

RESOURCE WORKSHOP

USER'S
GUIDE

B O R L A N D

Resource Workshop[™]

User's Guide

BORLAND INTERNATIONAL, INC. 1800 GREEN HILLS ROAD
P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001

Copyright © 1991 by Borland International. All rights reserved. All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Windows, as used in this manual, shall refer to Microsoft's implementation of a windows system.

PRINTED IN THE USA.

20 19 18 17 16 15 14 13

C O N T E N T S

Introduction	1	Bitmap files	21
Resource Workshop features	1	Icon files	21
Hardware and software requirements	2	Cursor files	21
What's in the manual	3	Font files	21
Conventions used in this manual	4	Identifier files	22
Menu commands	5	Using C header files	22
Typography and symbols	5	Using Pascal units and include files	23
How to contact Borland	5	.DRV files	24
Bibliography	6	How all these files work together— a sample project	24
Chapter 1 Getting started	9	Managing all the project files—the project window	27
Installing Resource Workshop	9	Tips for the new user	28
Starting Resource Workshop	9		
Using command-line options	10	Chapter 3 Working with projects and resources	31
Exiting from Resource Workshop	11	Creating a new project	32
Accessing Help	12	Opening an existing project	34
		Using the Project window	38
Chapter 2 Resource Workshop basics	13	Displaying information in the Project window	38
Understanding Windows resources	13	By File	38
Types of resources	14	By Type	39
Dialog boxes	14	Show Identifiers	39
Menus	15	Show Resources	39
Accelerators	15	Show Items	40
String tables	16	Show Unused Types	40
Bitmaps	16	Selecting a resource	40
Icons	17	Working with resources	40
Cursors	17	Editing a resource	40
Fonts	18	Using a resource editor	41
User-defined and rcdata resources	18	Using the internal text editor	41
The two types of editors	19	Adding a resource	42
Types of resource files	20	Adding a resource stored in a file	42
Resource compiler files	20	Existing resource files	42
Resource files	20	New resource files	44
Executable and dynamic link library files	21		
Dialog files	21		

Creating a new resource	44	Selecting multiple controls	73
Renaming a resource	45	Moving and resizing controls	74
Specifying resource memory options ..	47	Aligning controls with a grid	75
Removing a resource	49	Changing the appearance of	
Using identifiers	49	controls	76
Adding an identifier file	50	Giving a control a caption	77
Working without an identifier file	50	Changing a control's class	78
Adding identifiers by using a resource		Specifying which controls are tab	
editor	50	stops	78
Adding, editing, deleting, and listing		Tab-Set tool	79
identifiers	51	Style dialog box	79
Saving resources, files, and projects	53	Set Tabs command	79
File Save Project	53	Grouping related controls	80
File Save File As	54	Reordering the controls	80
Resource Save Resource As	54	Editing groups of controls	81
Configuration preferences	55	Aligning multiple controls	82
Undo Levels	55	Resizing multiple controls	85
Text Editor	56	Placing controls in columns and	
Include path	56	rows	86
Multi-Save	56	Undoing changes	88
.RES	56	Button controls	88
Executable	57	Scroll bar controls	90
Make backups when saving files ..	57	List box controls	90
Copying resources between projects	57	Edit text controls	93
Working with binary files	58	Static controls	95
Chapter 4 Creating dialog boxes	61	Iconic static controls	97
Starting the Dialog editor	62	Combo box controls	97
Creating a new dialog box	62	Custom controls	99
Editing an existing dialog box	64	Creating your own custom controls ..	99
Customizing a dialog box	64	Installing a custom control library ..	100
Defining a dialog box	64	Displaying custom controls	100
Adding a caption	65	Adding a custom control	100
Choosing the window type	66	Testing a dialog box	101
Choosing a frame style	66	Saving a dialog box	102
Specifying the dialog style	66	Saving the project	102
Specifying fonts	68	Saving a dialog box in a file	102
Including a menu	69	Viewing two dialog boxes	104
Assigning a custom class to a dialog		Customizing the Dialog editor	104
box	69	A sample project	107
Working with controls	70	Creating a new dialog box	109
Adding controls	72	Starting the Dialog editor	111
Adding multiple copies of a control ..	72	Customizing the dialog box	111
Editing controls	73	Adding the text control	111
		Adding the check boxes	112

Adding the push buttons	112	A sample menu	135
Testing the dialog box	113	Creating a menu using a text editor ..	135
Saving the project	113	Creating a menu using the Menu editor	136
Chapter 5 Creating menus	115	Creating the menu	136
The Menu editor screen	116	Adding commands to the Widgets menu	137
The Outline pane	117	Adding commands to the Arrange List menu	138
The Dialog Box pane	118	Testing the menu	138
The Test Menu pane	120	Chapter 6 Creating accelerators	139
Starting the Menu editor	121	Using the Accelerator editor	141
Creating a new menu	121	The Outline pane	141
Editing an existing menu	122	The Dialog Box pane	142
Customizing a menu	122	Starting the menu editor	143
Adding a new statement	122	Starting the Accelerator editor	144
Moving and copying	123	Creating a new accelerator table	144
Undoing errors	123	Editing an existing accelerator table .	144
Customizing a menu item	123	Customizing an accelerator table	145
New menu items	124	Selecting an accelerator key	145
Choosing where to insert a new menu item	124	Using the dialog box	146
Creating the menu item	126	Setting the command value	146
Selecting a menu item	126	Understanding ASCII and Virtual keys	146
Using the dialog box	127	ASCII keys	146
Entering item text	127	Virtual keys	147
Entering an item ID	127	Specifying the accelerator key	147
The remaining dialog box options	128	Manual mode	148
Customizing a pop-up command ...	128	Key Value mode	148
New pop-up commands	128	The flash feature	149
Choosing where to insert a new pop- up command	128	Moving and copying an accelerator ..	149
Adding a new pop-up command	129	Deleting an accelerator	149
Selecting a pop-up command	129	Undoing and redoing changes	149
Using the dialog box	129	Checking for duplicate key combinations	150
Entering item text	129	Editing a resource script for an accelerator table	150
The remaining dialog box fields .	129	Creating a sample accelerator table	151
Defining a menu separator	130	Chapter 7 Creating a string table	155
Deleting a menu statement	130	Starting the String editor	156
Testing a menu	131	Working with string tables	157
Saving changes	132	Windows and strings	157
Saving the project	132		
Saving the menu resource as a file ...	132		
Editing a menu resource script	133		

Entering a new string	158	Displaying a grid in a zoomed window	182
Editing existing strings	160	Reading the status line	183
Changing a string	160	The right side: current paint tool information	183
Restoring changed string values ..	160	The left side: menu command explanations	183
Deleting a string	160	Working with colors	184
Editing the resource script of a string table	160	Choosing the number of colors for a resource	184
Changing the string	160	Using a foreground color	185
Changing the identifier	161	Using a background color	186
Saving a string table	161	Including transparent and inverted areas in a cursor or icon	186
Testing a string table	162	Using transparent and inverted attributes	186
Creating a sample string table	162	Showing and hiding the Colors palette	187
Chapter 8 Using the Paint Editor	165	Customizing colors	188
Starting the Paint editor	165	Editing colors in the Colors palette ..	188
Loading cursors, fonts, and bitmaps .	165	The palette index	188
Loading icons	166	Editing a color	189
Features available to resources	167	Default button	190
Understanding foreground and background colors	168	System button	190
Using the Tools palette	168	Changing colors of transparent and inverted areas	190
The Pick Rectangle tool	170	Adding text to a resource	190
The scissors	171	Aligning text	191
The Zoom tool	171	Choosing fonts, size, and text style ..	192
Zooming the entire image	171	Choosing brush shapes	193
Zooming a portion of the image ...	171	Choosing paint patterns	194
The eraser	172	Choosing a line style	195
The pen	172	Aligning a selected area	196
The paintbrush	173	Resizing a selected area	196
The airbrush	173	Setting global Paint editor options	198
The paint can	174	Draw on both images	198
The Line tool	175	Grid on zoomed images	198
The Text tool	175	Save with default device colors ...	198
Painting empty frames	176	Chapter 9 Creating icons	199
Painting filled-in frames	177	Starting the Paint editor	200
The Hand tool	177	Creating a new icon	201
The style selections	178	Selecting a storage format	201
Using the two window panes	178	Binary format	202
Zooming	180		
Using the zoom accelerators	181		
Seeing which part of the image is zoomed	181		
Moving a zoomed image around	182		

Source format	203
Editing an existing icon	204
Customizing an icon	204
Design issues	204
Zooming the icon	205
Working with transparent and inverted areas	205
The background color	206
Changing the color of transparent and inverted areas	206
Making an icon look three- dimensional	207
Testing an icon	208
Saving an icon	209
Saving the project	209
Saving the icon resource as a file	210
Adding an image to an icon resource ..	211
Deleting icons and images	212
Deleting an icon resource	212
Deleting an icon image	212
Editing an icon resource script	213
Creating a sample icon	213
Creating the new icon	214
Changing the transparent color	214
Drawing the calculator	214
Adding a three-dimensional effect ...	216
Drawing the ledger page	217
Copying the image to a different color format	218
Chapter 10 Creating cursors	221
Starting the Paint editor	222
Creating a new cursor	223
Source format	224
Binary format	224
Editing an existing cursor	225
Customizing a cursor	225
Design issues	225
Zooming the cursor	226
Working with transparent and inverted areas	226
The background color	226
Changing the color of transparent and inverted areas	227

Setting the cursor's hot spot	228
Testing the cursor	229
Saving changes	229
Saving a project	230
Saving a cursor resource as a file	230
Editing a cursor resource script	231

Chapter 11 Creating bitmaps	233
Starting the Paint editor	234
Creating a new bitmap	234
Editing existing bitmaps	235
Customizing a bitmap	236
Changing bitmap size and colors	236
Saving a bitmap	238
Saving the project	238
Saving the bitmap as a file	238
Testing the bitmap	239

Chapter 12 Creating fonts	241
Starting the Paint editor	243
Creating a new font resource	243
Editing an existing font resource	244
Customizing a font resource	244
Changing a font image	245
Defining the character set for a font ..	245
Creating variable-width fonts	247
Defining a header for a font resource ..	249
Changing size and attributes	251
Deleting a font image	251
Saving a font resource	252
Saving the project	252
Saving a font resource as a file	252
Adding font resources to your application	253
Creating a .FON file with Turbo Pascal	253
Creating a .FON file with C++	253
Testing the font	254
A sample font resource	254

Chapter 13 Creating user-defined resources	257
Creating a resource type	258
Adding a user-defined resource	258
Editing a user-defined resource	260

Entering data in the resource script ..	261
Handling data stored in a separate file	262
Testing a user-defined resource	262
Using the RCDATA resource type	262
Deleting a user-defined resource	263

Appendix A Technical notes	265
Compiler differences	265
Dialog boxes as child or overlapped windows	266
Index	267

1.1: Resource Workshop command-line options	11
3.1: Resource Memory Options	48
4.1: Window types	66
4.2: Frame styles	66
4.3: Dialog box styles	67
4.4: Grid Type options	76
4.5: Options common to Style dialog boxes	76
4.6: Control attributes	77
4.7: Horizontal alignment options	83
4.8: Vertical alignment options	83
4.9: Horizontal size options	86
4.10: Vertical size options	86
4.11: Button types	88
4.12: Justification options	90
4.13: Alignment options	90
4.14: Owner Drawing options	91
4.15: List Box options	92
4.16: Justification options	93
4.17: Automatic Scroll options	94
4.18: Line options	94
4.19: Case options	94
4.20: Other options	95
4.21: Control Type options	96
4.22: Combo box type options	98
4.23: Owner drawing options	98
4.24: Combo box attributes	99
4.25: Status line units	105
4.26: Selection Border options	105
4.27: Drawing Type options	106
4.28: Selection options	106
5.1: Menu editor dialog box selections ..	119
5.2: View menu selections	120
6.1: Accelerator editor dialog box selections	142
8.1: Zoom commands	181
11.1: Bitmap size and attribute options ..	237
12.1: Font size options	246
12.2: Character options	247
12.3: Font header options	250
12.4: Font attributes	250
12.5: Font sizes	251

F I G U R E S

1.1: Empty Resource Workshop window ..10	3.15: The Rename Resource dialog box ...46
2.1: A typical dialog box14	3.16: The New Identifier dialog box47
2.2: A typical file menu15	3.17: The Resource Memory Options dialog box48
2.3: An open edit menu, with accelerators16	3.18: The New Identifier dialog box on the String Table editor51
2.4: The paintbrush bitmap16	3.19: The Identifiers dialog box52
2.5: The RWPDEMO icon17	3.20: The Save File As dialog box54
2.6: Paint editor with customized cursor ..18	3.21: The Preferences dialog box55
2.7: The dialog editor19	3.22: The Paste Resource dialog box58
2.8: MYPROJ.RC, the central project file ..25	4.1: A typical dialog box61
2.9: MYPROJ.RC points to .CUR and .BMP files25	4.2: The New Resource dialog box63
2.10: MYPROJ.RC points to .CUR, .BMP, and .H files26	4.3: The Dialog editor with an empty dialog box63
2.11: MYPROJ.RC bound into executable file26	4.4: The Window Style dialog box65
2.12: A typical project window27	4.5: The Select Font dialog box68
3.1: MYPROJ.RC32	4.6: The Tools palette70
3.2: Project window for MYPROJ.RC32	4.7: The Duplicate Control dialog box73
3.3: The New Project dialog box33	4.8: Dialog box coordinates75
3.4: Project Window with Save Project selected34	4.9: The Set Grid Attributes dialog box ...75
3.5: The Open File dialog box35	4.10: The Generic Control Style dialog box78
3.6: Compile Status dialog box for rwpdemo.rc36	4.11: The Align Controls dialog box82
3.7: Compile Error dialog box for rwpdemo.rc37	4.12: The Alignment palette84
3.8: The Project window37	4.13: The Size Controls dialog box85
3.9: The View menu38	4.14: The Form Controls Into An Array dialog box87
3.10: Project window showing resources by file39	4.15: Controls in left to right order87
3.11: Project window showing resources by type39	4.16: Controls in top to bottom order88
3.12: The Add File to Project dialog box ..43	4.17: A list box in a File Open dialog box .91
3.13: The New Resource dialog box44	4.18: An edit text control from a File Open dialog box93
3.14: New Icon Image dialog box45	4.19: A combo box from a File Open dialog box97
	4.20: The New Control Custom dialog box101

4.21: The .RC file points to .DLG and .H files	102	8.8: Zoomed image and dotted outline on unzoomed image	181
4.22: A project window, with rcinclude for a .DLG file	103	8.9: A zoomed image with a grid overlay	182
4.23: The Preferences dialog box	105	8.10: The 16-color Colors palette	184
4.24: The Project window for RWPDEMO	107	8.11: The New Bitmap Attributes dialog box	185
4.25: The sample Open dialog box	108	8.12: Colors palette with Transparent and Inverted colors	187
4.26: The sample dialog box	110	8.13: The Edit Color dialog box	188
5.1: A typical File menu	115	8.14: The 16-color palette index	189
5.2: The Menu editor with RWPDEMO's menu	117	8.15: An icon that includes text	190
5.3: A newly-created menu	120	8.16: Aligning text	191
5.4: The Menu editor window in alternate format	121	8.17: The Select Font dialog box	192
5.5: The RWPDEMO Project window	122	8.18: The Brush Shape dialog box	193
5.6: The Menu menu	123	8.19: The Set Pattern dialog box	194
5.7: The Widgets menu	124	8.20: The Set Pen Style dialog box	195
5.8: Script outline for the Widgets menu	125	8.21: The Align Selection dialog box	196
5.9: Menu with new items inserted	126	8.22: The Stretch Selection dialog box	197
5.10: The Save Resource As dialog box for a menu resource	133	8.23: The Set Paint Editor Options dialog box	198
5.11: The sample menu	135	9.1: The RWPDEMO icon	199
6.1: An Edit menu with accelerators	139	9.2: The Paint Editor with the RWPDEMO icon loaded	199
6.2: RWPDEMO accelerator table in Accelerator editor	140	9.3: Dialog box prompting for source or binary	201
6.3: RWPDEMO project window	145	9.4: The Add File To Project dialog box	202
7.1: The New Resource dialog box with STRINGTABLE highlighted	156	9.5: The New Icon Image dialog box	203
7.2: The String editor with string table entries	157	9.6: Paint editor window for a new 16-color icon	203
7.3: The New Identifier dialog box	163	9.7: The Set Transparent Color dialog box	206
7.4: The String editor with four strings defined	164	9.8: A black box icon with gray drop shading	207
8.1: The RWPDEMO cursor	166	9.9: The Save File As dialog box for an icon resource	210
8.2: Icon Window for the RWPDEMO icon	166	9.10: The Icon window	211
8.3: The Paint Editor Tools palette	169	9.11: Source script for an icon	213
8.4: Grabbing a zoomed image	178	9.12: Beginning screen for a new icon	215
8.5: The Paint Editor's split screen	179	9.13: Zoomed image of calculator before adding drop shading	216
8.6: Zoomed image taking entire edit area	179	9.14: Zoomed image of calculator with shading	217
8.7: The View menu	180	9.15: The Home Budget icon	218

10.1: Paint Editor with Paint Can cursor	221	11.3: The New Bitmap Attributes dialog box	235
10.2: The Paint Editor, with a simple cursor	222	11.4: The Set Bitmap Attributes dialog box	237
10.3: Dialog box prompting for source or binary	223	12.1: A customized bomb character	242
10.4: The Add File to Project dialog box	224	12.2: Dialog box prompting for source or binary	243
10.5: Edit Transparent and Inverted Colors dialog box	227	12.3: The Paint editor as it looks for a new font	244
10.6: The Set Hot Spot dialog box	228	12.4: Font characters in the Paint editor	245
10.7: Save File As dialog box for a cursor resource	230	12.5: The Font Size Information dialog box	246
10.8: Cursor resource script in Edit window	231	12.6: The Character Width dialog box	248
11.1: The Brush bitmap from the Paint Editor tool box	233	12.7: The Font Header Information dialog box	249
11.2: Dialog box prompting for source or binary	234	12.8: RWPDEMO font in the Paint editor	255
		13.1: The New Resource dialog box	259

Resource Workshop is a sophisticated tool that integrates the entire process of designing and compiling resources for applications running under Microsoft Windows, Version 3.0 and later. This manual includes chapters describing all the tools available in Resource Workshop and how to use them, as well as some general information on designing and using resources in Windows applications.

If you write applications that run under Windows, or if you want to modify the visual interface of Windows applications written by others, Resource Workshop is the easiest and most powerful way to get your programs looking the way you want.

Resource Workshop features

Resource Workshop provides everything you need to create and modify Windows resources, including

- Graphics-oriented visual resource editors that make it easy to design and modify resources.
- A text editor for manipulating text descriptions of resources.
- A compiler that lets you compile your resources incrementally and provides almost complete compatibility with the Microsoft Resource Compiler.

In short, Resource Workshop is a design tool that gives you everything you need to design all resources for your Windows programs. Among other things, Resource Workshop

- Is a flexible tool that works with resources in either text or binary format. It includes powerful graphics-oriented editors that let you edit binary files and a text editor that lets you edit the files as resource scripts.

- Makes it easy to manage hundreds of resources stored in dozens of files.
- Performs mundane tasks for you, such as automatically loading the correct editor when you choose a resource, inserting references to resource files as necessary in your .RC file, and adding **#defines** or constants for your resource IDs to the appropriate files.
- Includes all the compilers you need and makes it easy to compile your resources only when you need to.
- Decompiles binary resource files, allowing you to make changes to a program's resources even if you don't have access to the source code.
- Includes features that automatically check for errors, making it easy to test resources for errors like incorrect syntax and duplicate resource IDs.
- Is easy to use, allowing even those with limited programming experience to take part in designing user interfaces for application programs.
- Includes extensive, multilevel Undo and Redo features that let you step back through changes you have made.

Hardware and software requirements

The following are the minimum requirements for using Resource Workshop:

- You must have a computer capable of running Windows in standard or 386 enhanced mode (80286 processor or higher). The computer must have at least 2MB of RAM and a graphics display and adapter (Hercules, EGA, VGA, or better). A mouse or other pointing device is also required.
- Windows 3.0 or later must be installed on your computer.
- You must have at least 2.5MB of free disk space (3.5MB if you load all the files, including sample programs).

What's in the manual

This manual explains how to use Resource Workshop to develop Windows resources. It doesn't tell you how to write Windows programs or how to write code in your programs to access resources. The manual assumes that you know the basics of Windows programming, or that you are learning those basics in other books (see the bibliography on page 6).

The first part of this manual explains about Windows resources and how Resource Workshop manages them in projects.

- Chapter 1, "Getting started," describes how to install, start, and exit Resource Workshop and how to access the Help system.
- Chapter 2, "Resource Workshop basics," gives a brief introduction to the different kinds of resources available under Windows, the kinds of editors used in Resource Workshop to edit them, and the different kinds of files you can store the resources in. It also introduces the notion of a project, which includes all the resources for a given program.
- Chapter 3, "Working with projects and resources," covers projects in more detail, describing how to set up and use projects, edit and add resources, and coordinate the identifiers used in your resources with those in your program.

The remaining chapters describe the different resource editors contained in Resource Workshop.

- Chapter 4, "Creating dialog boxes," covers the Dialog editor, including all aspects of creating and modifying dialog boxes and the controls they contain. The Dialog editor enables you to design, modify, and test your dialog boxes outside your program.
- Chapter 5, "Creating menus," describes the use of the Menu editor, a visual tool that lets you create, modify, and test pull-down and pop-up menus right on the screen.
- Chapter 6, "Creating accelerators," describes the editor that creates accelerators, the key-combination shortcuts for commands in your Windows applications. With the Accelerator editor you can bind keystrokes to your application's commands, either by describing the keys you want to bind or by actually typing the keys.
- Chapter 7, "Creating a string table," describes how to use the String editor, which lets you create and maintain string table

resources. These resources contain groups of strings that are the text used by your program for error messages and prompts.

- Chapter 8, “Using the Paint editor,” provides the basics for working with the Paint editor, which Resource Workshop starts when you choose a bitmapped resource (an icon, a cursor, a bitmap, or a font). This chapter focuses on the Paint editor features common to all the bitmapped resources; the chapters that follow describe the specifics of working with each particular kind of bitmapped resource.
- Chapters 9 through 12 describe in detail how to edit icons, cursors, bitmaps, and fonts.
 - Icons (Chapter 9, “Creating icons,”) are the small, square images used to represent programs and windows that have been minimized.
 - Cursors (Chapter 10, “Creating cursors,”) are the images used to indicate the position of the mouse pointer on the screen.
 - Bitmaps (Chapter 11, “Creating bitmaps,”) are various graphical images used for many purposes, including backgrounds, pictures, and special controls.
 - Fonts (Chapter 12, “Creating fonts,”) are the images used to represent text characters in windows. Fonts can also be collections of bitmaps stored in a more economical format.
- Chapter 13, “Creating user-defined resources,” explains how to use any other kinds of resources you might want to define. All the resources described in earlier chapters are the standard ones defined and handled by Windows. If these resources don’t meet your needs, you can use the resource mechanism to create user-defined resources that store other kinds of resource data for your programs.
- Appendix A, “Technical notes,” provides technical notes on a number of aspects of Resource Workshop, including compatibility with the Microsoft Resource Compiler, use of dialog boxes as child windows, and use of custom controls.

Conventions used in this manual

There are a number of conventions used in this manual to represent things like keys, menu commands, program source code, and language identifiers.

Menu commands

This manual uses a shorthand when referring to a series of menu commands. Instead of saying, "Choose the Cut command from the Edit menu," it says instead, "Choose the Edit | Cut command."

Typography and symbols

This manual uses the following symbols and special fonts:

Monospace type	This typeface represents text as it appears on-screen or in a program. It is also used for anything you must type.
[]	Square brackets in text or DOS command lines enclose optional items that depend on your system. <i>Text of this sort should not be typed verbatim.</i>
Boldface	This typeface is used in text for Turbo Pascal and Borland C++ reserved words and for Borland C++ function, class, and structure names.
<i>Italics</i>	Italics indicate identifiers that appear in text. They can represent terms that you can use as is, or that you can think up new names for (your choice, usually). They are also used to emphasize certain words, such as new terms.
Keycaps	This typeface indicates a key on your keyboard. For example, "Press Esc to exit a menu."

How to contact Borland

You can also leave messages on BPROGA, the Pascal forum.

The best way to contact Borland is to log on to one of Borland's forums on CompuServe: Type GO BPROGB from the main CompuServe menu to get on Borland Programming Forum B (Borland languages and tools) from the Borland main menu. Leave your questions or comments there for the support staff to process.

If you prefer, write a letter with your comments and send it to

Borland International
Technical Support Department – Resource Workshop
1800 Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95067-0001, USA

408-438-5300 You can also telephone our Technical Support department. Please have the following information handy before you call:

1. Product name and serial number on your original distribution disk. Please have your serial number ready, or we won't be able to process your call.
2. Product version number. The version number for Resource Workshop is displayed when you first load the program and before you press any keys. If you are in Resource Workshop, choose About from the Help menu.
3. Computer brand, model, and the brands and model numbers of any additional hardware.
4. Operating system and version number. (The version number can be determined by typing `VER` at the MS-DOS prompt.)
5. Contents of your `AUTOEXEC.BAT` file.
6. Contents of your `CONFIG.SYS` file.

Bibliography

There is a considerable amount of information available on Windows programming in the manuals that come with your Borland language compiler. In addition, you might find the following books helpful in designing and using Windows resources.

Microsoft Windows Software Development Kit, Guide to Programming, Microsoft Corporation. (Redmond, WA: 1990).

Microsoft Windows Software Development Kit Reference, Vols. 1 and 2, Microsoft Corporation. (Redmond, WA: 1990).

Microsoft Windows Software Development Kit, Tools, Microsoft Corporation. (Redmond, WA: 1990).

Petzold, Charles. *Programming Windows*, Microsoft Press. (Redmond, WA: 1990).

Systems Application Architecture Common User Access Advanced Interface Design Guide, International Business Machines Corporation. (1989).

Getting started

This chapter gives you information about how to install Resource Workshop and covers the basics for starting, exiting, and getting Help in Resource Workshop.

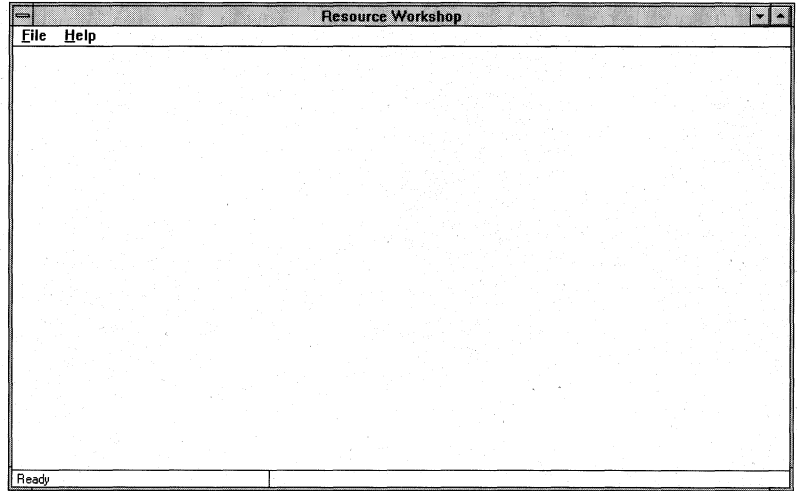
Installing Resource Workshop

To install Resource Workshop and its sample files and programs, run the INSTALL program on your installation disk. For more information about installation, see the README file on the same disk.

Starting Resource Workshop

Once you've installed Resource Workshop and it appears as an icon in a Windows program group, double-click the icon to start Resource Workshop and display the Resource Workshop main window.

Figure 1.1
Empty Resource Workshop
window



You can use the File menu to create a new project or open an existing project. See Chapter 3 for more details.

Using command-line options

There are a number of command-line options you can use when you start Resource Workshop. You can enter these command-line options in three ways:

- Start Windows and Resource Workshop at the same time from the DOS prompt (for example, `WIN WORKSHOP -X`).
- Use the Program Manager's File | Run command.
- Use the Program Manager's File | Properties command to change the Command Line property of the Resource Workshop icon.

The format for command-line options is as follows:

```
WORKSHOP [option [optionarg] ... option [optionarg]]
```

- An *option* is one of the command-line switches in table 1.1. It must be preceded by a dash (-) or a slash (/) and can be any combination of uppercase and lowercase letters.
- An *optionarg* is the argument to an option, such as the path name that follows the `-i` option.

Table 1.1
Resource Workshop
command-line options

The following table lists the Resource Workshop command-line options:

Option	Description
-x	Clears the include path. If you've set an include path using the Preferences dialog box (see page 55), this option erases your settings.
-i <i>pathname</i>	Adds an additional path specification to the include path. For example: <pre>WORKSHOP -i c:\mystuff\include</pre>
-fo <i>filename</i>	Sets the .RES Multi-Save option (see the Preferences dialog box on page 55) to the specified file name. With this option set, whenever Resource Workshop compiles the current project, it also saves the resources in binary format to the indicated .RES file. For example: <pre>WORKSHOP -FO C:\MYSTUFF\MYPROJ.RES</pre>
-fx <i>filename</i>	Sets the Executable Multi-Save option (see the Preferences dialog box on page 55) to the specified file name. With this option set, whenever Resource Workshop compiles the current project, it also binds the resources in binary format to the indicated .EXE file. For example: <pre>workshop /fx c:\mystuff\myproj.exe</pre>

Exiting from Resource Workshop

When you finish working with Resource Workshop, choose File | Exit or double-click the Control-menu icon at the top left corner of the Resource Workshop window.

If you've made any changes you haven't saved, Resource Workshop asks if you want to save the changes before quitting.

Accessing Help

You can access Help in Resource Workshop just as you can in any other Windows application:

- Press *Alt+H* or select the Help command on the main menu to bring up the Help menu.
- Press *F1* to display the Help index directly, without going through the Help menu.

In addition to standard Windows Help, Resource Workshop provides an additional Help feature, context-sensitive Help. There are two ways to access this feature:



- If you want to know more about a menu item you've selected, press *F1* to directly access Help for the item.
- If you press *Shift+F1*, the cursor changes into the Help cursor, a question mark with a cross hair (see the cursor on the left). To get context-sensitive Help, position the cross hair on the menu command or screen area you want to know more about and click the mouse.

Help runs as a separate application under Windows. You can leave Help running and return to Resource Workshop, or you can terminate Help altogether by pressing *Alt+F4*, double-clicking the Control-menu box, or choosing File|Exit.

Resource Workshop basics

This chapter provides an overview of Resource Workshop and Windows resources. It starts with a discussion of what resources are and introduces each resource type. It also covers how to use the different types of resource files in Resource Workshop and how these files fit together in a project. The end of the chapter lists some tips to get you started with Resource Workshop.

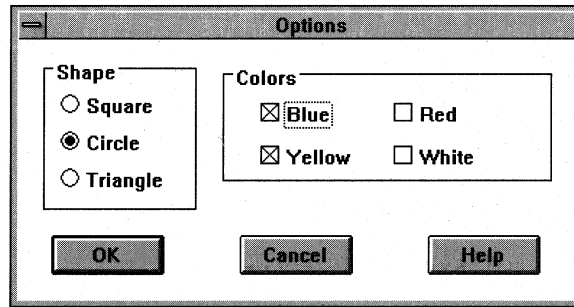
Understanding Windows resources

Resources are data that define the visible portions of your Windows program. For example, when you open a dialog box and click a button to accomplish a task with a program, you're interacting with that program's resources. In addition to dialog boxes and buttons, other types of resources you can use in your Windows programs include icons, cursors, bitmaps, menus, and keyboard accelerators.

In Windows applications, resources provide a consistent user interface that makes it easy for users to switch from one Windows program to another.

The following is an example of a resource—a dialog box that lets users make certain choices:

Figure 2.1
A typical dialog box



The entire dialog box and all the controls in it (including buttons, check boxes, and so on) are defined in the Windows program as resources.

In general, a Windows application's resources are separate from the program code, letting you make significant changes to the user interface without even opening the file that contains the program code.

For example, suppose you're working with a Windows financial application. You can store the code for your financial algorithms in a separate file and can even compile separately from the program's resources. To change the way the program looks, you (or someone else responsible for user interfaces) can modify existing resources and create new ones, without having to worry about affecting the underlying financial calculations.

Also, because different applications can use the same set of resources, you don't have to reinvent all your favorite dialog boxes, icons, and customized cursors. Instead, you can use them over and over.

Types of resources

Resource Workshop supports the following types of Windows resources:

Dialog boxes

See Chapter 4 for information on editing dialog boxes.

A *dialog box* is a window (usually a popup window) that communicates information to the user and lets the user select

choices, such as files to open, colors to display, text to search for, and so on.

See Figure 2.1 for a sample dialog box.

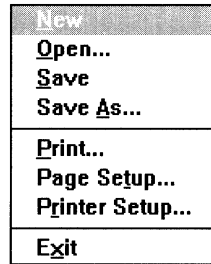
A dialog box usually includes *controls*, such as radio buttons, check boxes, push buttons, and so on. Resource Workshop makes it easy to put any combination of controls in a dialog box. Also, Resource Workshop lets you test your dialog box and debug its behavior before binding it to your executable code.

Menus

See Chapter 5 for information on editing menus.

Windows programs usually include a menu bar that lists user choices. Each menu item generally can display an additional list of choices. For example, most Windows programs include a File menu that includes menu commands for creating a new file or opening an existing file.

Figure 2.2
A typical file menu



Resource Workshop makes it easy to create menus by displaying the menu as you create it. You can test the menu as you add or modify menu items. Also, Resource Workshop writes text representing the menu on your screen as you use the Menu editor to create the menu.

Accelerators

See Chapter 6 for information on editing accelerators.

Accelerators are keyboard combinations (or hotkeys) that a user presses to perform a task in an application. For example, a Windows program can include the accelerator *Shift+Ins*, which the user presses to paste text or images from the clipboard into a file the program has open.

Figure 2.3
An open edit menu, with
accelerators

<u>U</u>ndo	Alt+Backspace
<u>C</u>ut	Shift+Del
<u>C</u>opy	Shift+Ins
<u>P</u>aste	Ctrl+Ins
<u>C</u>lear	
<u>D</u>elete	

In general, Windows programs use accelerators to duplicate menu functions, allowing the user to choose the function either by selecting from a menu or by pressing accelerator keys. However, you can also create accelerator resources that define new functions not available from your program's menus.

Resource Workshop makes it easy to create new accelerators and edit existing accelerators. Also, Resource Workshop writes the resource script text on your screen as you edit your accelerators so you can see exactly what's going on.

String tables

*See Chapter 7 for
information on editing string
tables.*

String tables contain text (like descriptions, prompts, and error messages) that's displayed as part of a Windows program. Because these text strings are Windows resources that are separate from the program (instead of strings embedded in the program), you or others can edit and translate messages displayed by a program without having to make any changes to the program's source code.

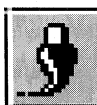
Bitmaps

*See Chapter 11 for
information on painting
bitmaps.*

A *bitmap* is a binary representation of a graphic image in a program. Windows itself uses lots of bitmaps. For example, the images representing various controls on a typical window, such as scroll bar arrows, the Control-menu symbol, and the Minimize symbol, are all bitmaps. Each bit, or group of bits, in the bitmap represents one pixel of the image.

The following figure shows an example of a bitmap: the Paintbrush tool from the Paint editor Tools palette.

Figure 2.4
The paintbrush bitmap



See Chapter 8 for a description of the Paint editor.

You create bitmaps using Resource Workshop's Paint editor, which lets you customize colors, choose from numerous paint tools, look at two different views of the bitmap you're creating, and so on. You also use this editor to work with the other specialized bitmapped resources: icons, cursors, and fonts.

Icons

See Chapter 9 for information on painting icons.

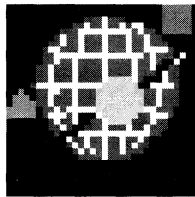
Icons can be transparent or inverted so that the background area shows through the image.

Icons are small bitmaps, generally 32×32 or 16×32 pixels in size, used to represent minimized windows. You create icons using the Paint editor (described in Chapter 8).

To see an example of a Windows icon, click the Minimize button of the Resource Workshop program to shrink the window. (The Minimize button is the down arrow symbol located in the upper right corner of the window.) As you can see, Windows uses the Resource Workshop icon to represent the minimized window.

The following icon comes with the demo project RWPDEMO.RC.

Figure 2.5
The RWPDEMO icon



Cursors

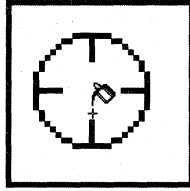
See Chapter 10 for information on painting cursors.

Cursors can be transparent or inverted so that the background area shows through the image.

A *cursor* is a small bitmap, 32×32 pixels in size, that represents the position of the mouse on the screen. Windows programs use customized cursors to indicate what type of task the user is currently performing. You create cursors using the Paint editor (see Chapter 8).

You can see an example of customized cursors in the Resource Workshop Paint editor, which comes up if you edit a bitmapped resource, like a cursor. Each time you choose a new paint tool and move the cursor to an image, the cursor takes a shape that represents its current function. For example, if you click the paint can in the Paint editor tool box and move the cursor to the image, the cursor becomes a paint can.

Figure 2.6
Paint editor with customized
cursor



Fonts

See Chapter 12 for
information on editing fonts.

Windows programs use *fonts* to define the typeface, size, and style of text. For example, a particular character that a program can display onscreen or print on a printer might be **10-point Times Roman bold**. In this case, the *typeface* is Times Roman, the *size* is 10 points, and the *style* is bold.

Another use of Font resources in Windows programs is to store a set of bitmaps that are always used together.

You can use Resource Workshop to modify the way existing fonts appear or create your own fonts.

User-defined and rcdata resources

See Chapter 13 for
information on editing user-
defined resources.

User-defined resources and *rcdata* resources (essentially the same in Resource Workshop) consist of any data you want to add to your executable file. For example, if you have a large block of initialized, read-only data, such as a text file, you can add it to your executable file as a user-defined resource.

One common reason for adding user-defined resources to an application is to help manage memory. Many Windows applications use the medium memory model, which includes a single data segment. If you have a relatively large amount of data and you don't want that data to take up permanent residence in memory, you can save the data as a user-defined resource defined to be discardable data. The user-defined resource then takes up memory only when your program needs to use it.

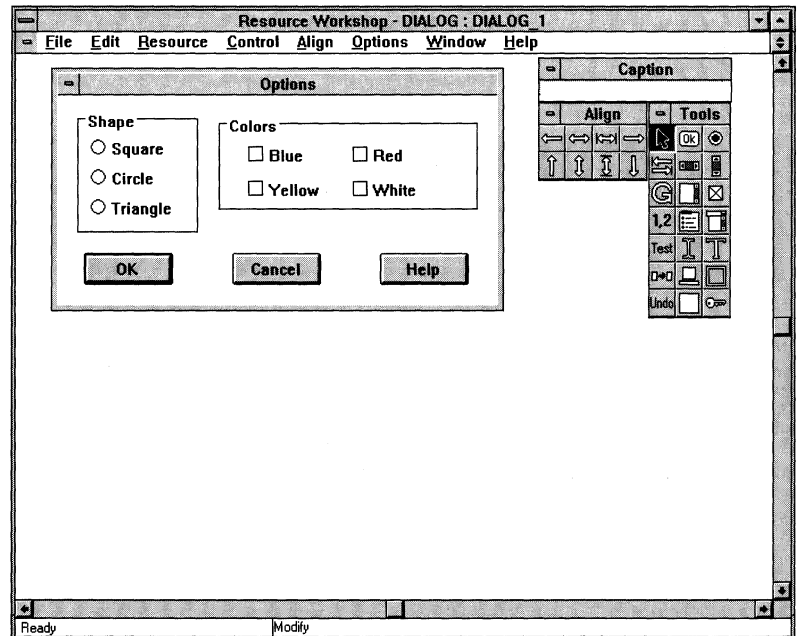
The two types of editors

Resource Workshop also lets you specify an external editor to use for editing project files that won't compile when they're being opened. See page 56 for more information.

Resource Workshop provides you with powerful, easy-to-use, graphics-oriented editors that you can use to design resources. However, Resource Workshop also gives you the option of using a text editor with resource scripts to design resources. This option makes Resource Workshop a flexible product accessible to all levels of Windows programmers.

For example, even if you're new to Windows programming, you can quickly design a dialog box using the Dialog editor. Creating and moving controls is as easy as clicking and dragging.

Figure 2.7
The dialog editor



Instead of a resource editor, you might want to use Resource Workshop's internal text editor for several reasons:

- If you're an experienced Windows programmer, you might prefer to work directly with resource scripts.
- You might find that some things are easier to fix with a text editor, while major changes are easier to make with the resource editor.

- You might want to use a text editor to take advantage of some minor capabilities not supported by the resource editor, such as rarely used options or a particular flag.

However, even experienced programmers find that the resource editors are fast and powerful, and free you to be more flexible in creating your resources.

Types of resource files

Not all Resource Workshop .RC files will be compatible with the RC compiler. Compile these files to .RES format with Resource Workshop.

A file you create and edit with Resource Workshop can be in either binary or text format. In addition, Resource Workshop can generate standard Windows file formats, which means you can use Resource Workshop files with any programs that generate binary code from resource script files, such as the Microsoft Resource Compiler.

Resource compiler files

See the on-line Help for information on the script commands.

A resource compiler (.RC) file is a resource script (text) file containing definitions of one or more resources. The file can contain resources defined in script form and references to other files containing resources.

In general, you should base all your Resource Workshop projects on at least one .RC file.

Resource files

A resource (.RES) file contains one or more compiled resources.

Typically, when creating a Windows program, you compile all resources for an application into a single .RES file, and then you bind the .RES file to the executable file as part of the linking process. However, with Resource Workshop if you don't want to produce a .RES file, you don't have to, because Resource Workshop can compile resource files and bind them directly to an executable file.

Executable and dynamic link library files

An executable (.EXE) or dynamic link library (DLL) file is the ultimate destination for all resources you define with Resource Workshop. Usually, you compile an .RC file into a .RES file, then use your compiler to bind the .RES file to the executable or DLL file. You can also use Resource Workshop to bind the resources directly to the executable or DLL file and bypass the RC compiler altogether.

If you want to change the resources in a compiled binary file (an executable file, a DLL file, or a .RES file), Resource Workshop will decompile the file and let you make changes, and then save the resources back to the original binary file.

Dialog files

A dialog (.DLG) file is a resource script (text) file that typically contains descriptions of one or more dialog boxes. There is no requirement that these resources be dialogs; they can be any resources found in an .RC file.

Bitmap files

A bitmap (.BMP) file contains a Bitmap resource in binary format.

Icon files

An icon (.ICO) file contains an icon in binary format.

Cursor files

A cursor (.CUR) file contains a customized cursor in binary format.

Font files

Font files take two forms, binary and font library.

- A binary font (.FNT) file contains the definition of a customized font in binary format. You can use the Resource Workshop Paint editor to design a font and store it in an .FNT file.
- A font library (.FON) file is a resource-only dynamic link library that contains a font directory and one or more fonts. You

See page 255 for information on .FON files.

must create .FON files outside Resource Workshop. However, once you've created a .FON file, you can use Resource Workshop to modify the file.

Identifier files

For more information on using Resource Workshop with identifiers, see page 49.

Windows programs use numbers to uniquely identify each resource. You use these numbers in your programs to identify the resources you want to work with. However, numbers are not very descriptive. For applications written in C, it's common to use **#defines** to assign symbolic definitions to constant values. Applications written in Pascal use constant declarations.

➡ This manual uses the term *identifier* to stand for both **#defines** and constants.

With identifiers, you create meaningful names to take the place of not-so-meaningful numbers. For example, although you could use the number 100 to uniquely identify a menu, an identifier like FILEMENU is much more descriptive.

If you're writing a C program, you store your resource **#defines** in a header (.H) file. For a Pascal program, you store your resource constants in a unit (.PAS) or include (.INC) file.

➡ This manual refers to all three kinds of files as *identifier files*.

Using C header files

If your program is written in C, you use **#defines** and store them in a header file. The following code shows a typical header file:

```
/******  
*                               Defines for demo program BCWDEMO.                               *  
******/  
  
#define LINE 1  
#define ELLIPSE LINE + 1  
#define RECTANGLE LINE + 2  
  
#define MID_QUIT 100  
#define MID_LINE 201  
#define MID_ELLIPSE MID_LINE + 1  
#define MID_RECTANGLE MID_LINE + 2  
#define MID_THIN 301  
#define MID_REGULAR MID_THIN + 1  
#define MID_THICK MID_THIN + 2
```

In addition to **#defines**, you can also store type and structure definitions, program code, and comments in a header file.

Resource Workshop ignores all data in the header file except for the **#defines** and any preprocessor directives.

Using Pascal units and include files

If your program is written in Pascal, you can use constants and store them in either a unit or an include file. You can store only comments and compiler directives with the constants, and the compiler can handle only constants defined as untyped integer and long integer expressions.

The following code shows a typical unit:

```
{*****  
*                               Constant declarations for TDODEMO.                               *  
*****}  
  
unit TdodemoConst;  
  
interface  
  
const  
    LINE = 1;  
    ELLIPSE = LINE + 1;  
    RECTANGLE = LINE + 2;  
  
    MID_QUIT = 100;  
    MID_LINE = 201;  
    MID_ELLIPSE = MID_LINE + 1;  
    MID_RECTANGLE = MID_LINE + 2;  
    MID_THIN = 301;  
    MID_REGULAR = MID_THIN + 1;  
    MID_THICK = MID_THIN + 2;  
  
implementation  
  
end.
```

Besides comments and compiler directives, only constant declarations are allowed in the **interface** section. The implementation section must be empty. Anything can follow the **end** keyword, but the Resource Workshop compiler ignores it.

An include file can contain only constant declarations, comments, and compiler directives, as shown in the following example:


```

{*****
*
*          Constant declarations for TDDemo.
*
*****}

const
    LINE = 1;
    ELLIPSE = LINE + 1;
    RECTANGLE = LINE + 2;

    MID_QUIT = 100;
    MID_LINE = 201;
    MID_ELLIPSE = MID_LINE + 1;
    MID_RECTANGLE = MID_LINE + 2;
    MID_THIN = 301;
    MID_REGULAR = MID_THIN + 1;
    MID_THICK = MID_THIN + 2;

```

.DRV files

A .DRV file is a Windows device driver, a special case of a DLL. You can edit the resources in one of these files just as you can in any DLL.

How all these files work together— a sample project

Resource Workshop makes it easy to keep track of resources by organizing resources into *projects*. In very general terms, a project consists of at least one of the following types of resource files:

- A resource script (.RC or .DLG) file
- A binary resource (.RES) file
- A binary cursor (.CUR) file
- A binary icon (.ICO) file
- A binary bitmap (.BMP) file
- A binary font (.FNT) file
- A font library (.FON) file
- An executable (.EXE) file
- A dynamic link library (DLL) file
- A Windows device driver (.DRV) file

For example, if you wanted to create a Resource Workshop project consisting only of a cursor, you could create a Resource Workshop project that contains a single .CUR file.

To take full advantage of all the Resource Workshop functions, your Resource Workshop projects should be built around an .RC file. For example, a single .RC file could refer to multiple dialog boxes, cursors, icons, bitmaps, and fonts.

Here's an example of a set of files you might work with. First, suppose you create an .RC file and call it MYPROJ.RC. This file will be the central file in your project. You can add as many different types of resources as you want, but everything in your project will be referenced in your .RC file.

Figure 2.8
MYPROJ.RC, the central
project file

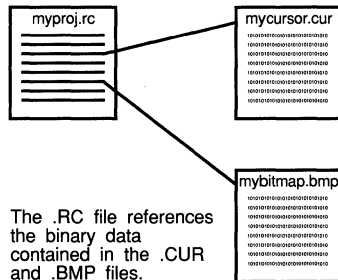


The .RC file is the central file in your Resource Workshop project.

Suppose you want to create a cursor and a bitmap. You can create these new resources using Resource | New and indicate that you want to store them in external files, the cursor in a .CUR file and the bitmap in a .BMP file. Resource Workshop puts references to the files in your .RC file, so that when you use the Resource Workshop Paint editor to draw the cursor and bitmap, Resource Workshop will store them in the appropriate files.

The following figure shows a diagram of these file connections:

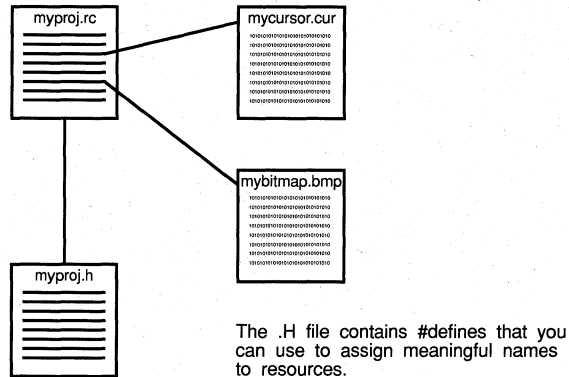
Figure 2.9
MYPROJ.RC points to .CUR
and .BMP files



The .RC file references the binary data contained in the .CUR and .BMP files.

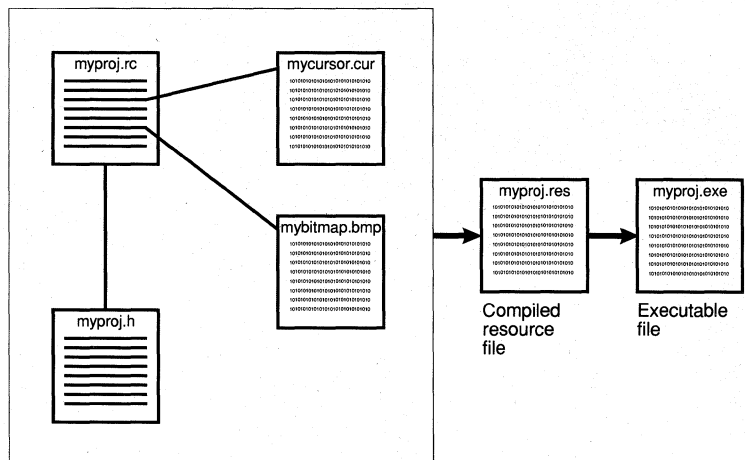
Now, suppose you want to use identifiers to assign meaningful names to resources. If you add a header (.H) file to the project, Resource Workshop puts a reference to the header file in the .RC file and is then able to store any new identifiers in the header file.

Figure 2.10
MYPROJ.RC points to .CUR,
.BMP, and .H files



So far, you've been working with resource data files, all of which you can compile into a single .RES file with Resource Workshop and then bind into an existing executable file with your Borland language compiler.

Figure 2.11
MYPROJ.RC bound into
executable file



Of course, your Resource Workshop projects will probably be considerably larger and more complex than the simple example provided here. And you can do some things differently, such as skipping the .RES file and compiling directly from your source data into an executable file. However, the concept of the project remains constant, and once you feel comfortable with the idea, creating your own Resource Workshop projects will be easy.

Managing all the project files—the project window

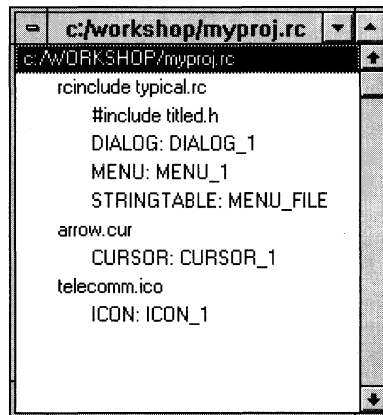
Resource Workshop makes it easy for you to design the resources you need and to manage all the resources for an application.

Whether you're creating a new project or working with an existing project, one of the first things you'll see in Resource Workshop is the *project window*. The project window shows you information about the project you're working with, including the types of resources and files that are included in the project.

Many projects consist of a main .RC file that contains references to other resources. The references in the main .RC file can include

- Other .RC files
- .DLG files
- Binary resource files, such as .BMP (bitmap), .ICO (icon), .CUR (cursor), and .FNT (font) files
- .H (header) files containing **#defines** that assign meaningful names to your resources
- .INC (include) or .PAS (unit) files containing Pascal constants that assign meaningful names to your resources

Figure 2.12
A typical project window



In addition to providing you with an overview of all of the files and resources contained in the project, Resource Workshop's project window also performs updates and recompiles when they are required. For example, if you change a resource identifier, Resource Workshop automatically recompiles all resources affected by the change.

The next chapter discusses the Project window in more detail and explains how to use its menus and features to set up and work with a project.

Tips for the new user

The most important concept to learn about Resource Workshop is an easy one: *the project*. Basically, organizing one or more resources into a project makes it easier to work with your resources and for Resource Workshop to take care of many details for you, such as placing references in the appropriate files. In most cases, you'll want to create projects based on an .RC file to take full advantage of all the Resource Workshop functions.

If you are a new Resource Workshop user, take a look at Chapter 3 for information about creating and managing a project. Reading Chapter 3 will give you the background you need to begin creating resources.

Here are some tips that can help you get started more quickly:

- Use the File menu to open existing projects or create new ones.
- Use the status line at the bottom of the Resource Workshop window and the online Help for explanations of Resource Workshop functions.
 - The left side of the status line provides you with a brief explanation of currently highlighted menu commands. The right side of the status line displays useful information when you use any of the resource editors.
 - The online Help provides more in-depth explanations of all aspects of Resource Workshop. You can also access context-sensitive Help by pressing *Shift+F1* to get the Help cursor and clicking the help cursor on the topic you want to know more about.
- If you're using identifiers, make sure you store them in the appropriate identifier file. It's best to avoid storing identifiers in your resource files. Whenever you store an identifier, Resource Workshop prompts you for the location in which to store it.
- Resource Workshop includes Undo and Redo features that let you step back and forth through previous tasks you have performed. For example, you can undo lines you have painted

See page 55 for information on setting preferences.

on a bitmap. Use File | Preferences to specify the maximum number of levels you can undo.

- You don't need access to source code to work with an application's resources. Resource Workshop can decompile the resources in an existing executable file to let you make changes to them.
- You can work with existing executable, DLL, and .FON files, but you cannot create a new executable, DLL, or .FON file with Resource Workshop.

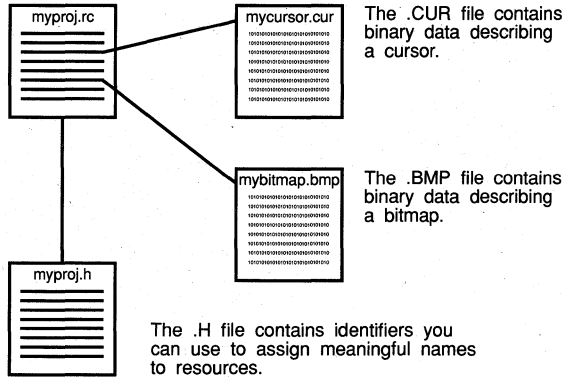
Working with projects and resources

This chapter describes the Project window. It explains how to control the display of resources in that window and how to choose resources from that window for editing. It also covers some general topics about resources, such as how to add them to and delete them from projects, how to save them, and how to work with identifiers.

As explained in Chapter 2, a *project* is a collection of one or more resources. A project is stored in a file that contains one or more resources, or that refers to files containing resources, or both. Typically this project file is a resource compiler (.RC) script file. Here's another look at the resource files for the sample project from Chapter 2:

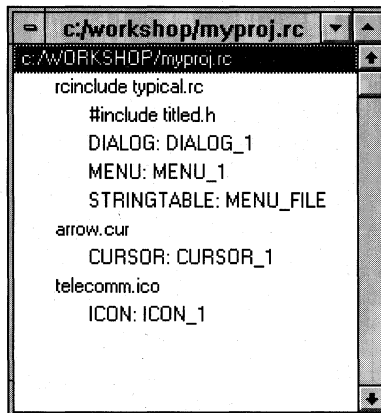
Figure 3.1
MYPROJ.RC

The .RC file is the central file in the project and references other files.



When you're working with a project, Resource Workshop displays it in the Project window. The following figure shows how the Project window looks for MYPROJ.RC if you're viewing resources grouped by file name:

Figure 3.2
Project window for
MYPROJ.RC

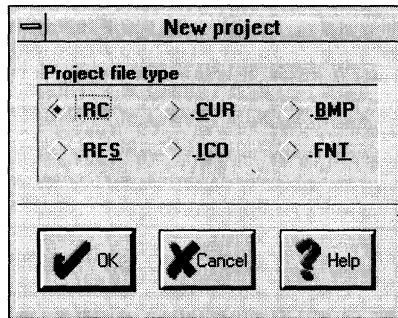


Creating a new project

To create a new project, do the following:

1. Choose File | New Project. Resource Workshop displays the New Project dialog box:

Figure 3.3
The New Project dialog box



Use this dialog box to choose the type of file on which to base your project. In most cases, you'll want to base projects on an .RC file because this type of file lets you add in any resource you want. However, you can also choose one of the following:

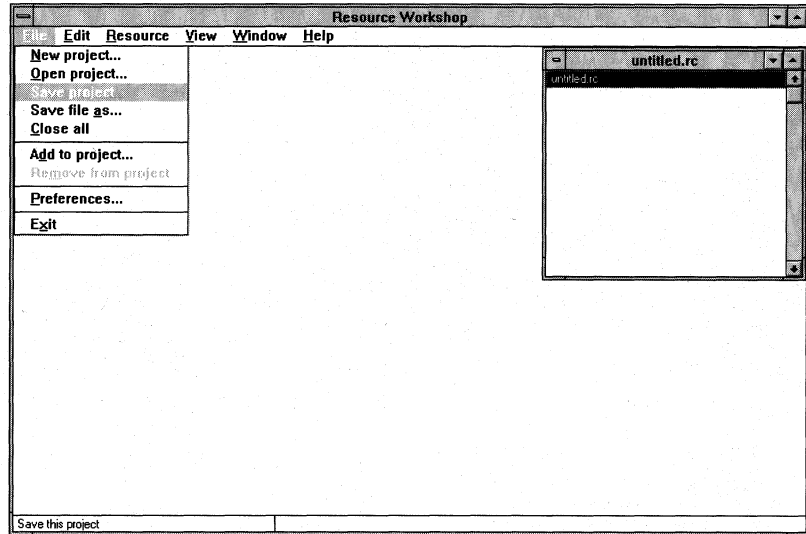
- .RES to work with a binary resource file
- .CUR to create a project containing a cursor
- .ICO to create a project containing an icon
- .BMP to create a project containing a bitmap
- .FNT to create a project containing only fonts

2. Click the project file type you want, then click OK.

(If you have a project open, Resource Workshop closes it first. If there are unsaved changes, before closing the project, Resource Workshop asks if you want to save those changes.)

Resource Workshop displays your new, untitled project in the Project window. You can name the project by using either of the File | Save commands; you can create new resources by using Resource | New; and you can add new or existing resources stored in other files by using File | Add To Project.

Figure 3.4
Project Window with Save
Project selected



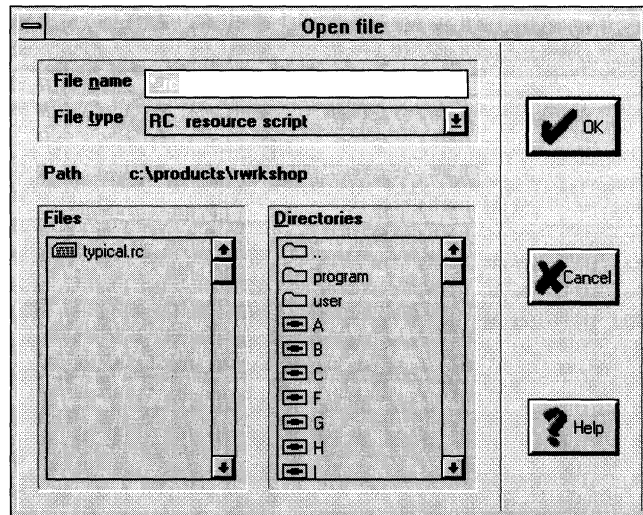
Opening an existing project

An existing project can be one that you created with Resource Workshop or an .RC file you created with other resource development software. You can also work with the resources in any application developed for Windows 3.0 or higher, even if you don't have access to the source code. If you have access only to an executable file, Resource Workshop can decompile the resources bound to that file so that you can make changes to them.

To open an existing project, do the following:

1. Choose File | Open Project. Resource Workshop displays the Open File dialog box.

Figure 3.5
The Open File dialog box



If you work with a decompiled binary file, you can't use identifiers: you must use only integer IDs with resources in these file.



To the right of File Type, click the list box arrow button to see all the different types of files you can open. If you're using Resource Workshop to develop resources, in most cases you'll want to work with .RC files. However, you can open any of the listed file types. If you open a binary file (such as an .EXE, a .CUR, an .ICO, or a .RES file), Resource Workshop decompiles the file to let you make changes.

If you type one of the standard extensions in the File Name text box (instead of picking a file type from the list and letting Resource Workshop insert the extension for you), Resource Workshop assigns the proper file type to the file. However, if you use a nonstandard extension (such as .MNU for a file you've stored a menu in), be sure to pick the correct file type from the File Type list before loading the file. (In the case of the .MNU file, if the resource is a menu stored as a resource script, the file type would be RC Resource Script.)

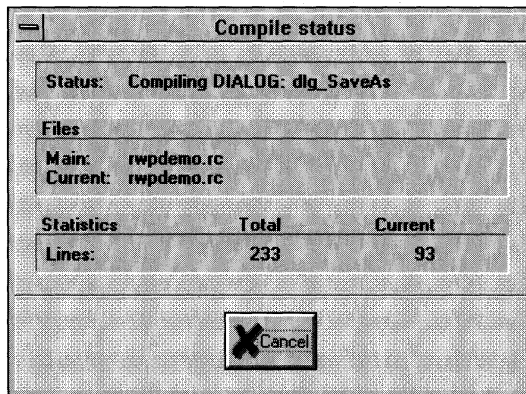
2. Specify the file containing the project you want to open by doing either of the following:

- Type the file name and press *Enter*.

If you type a file name, you must also specify a path if the file isn't in the current directory. For example, you might type C:\TEST\MYPROJ.RC.

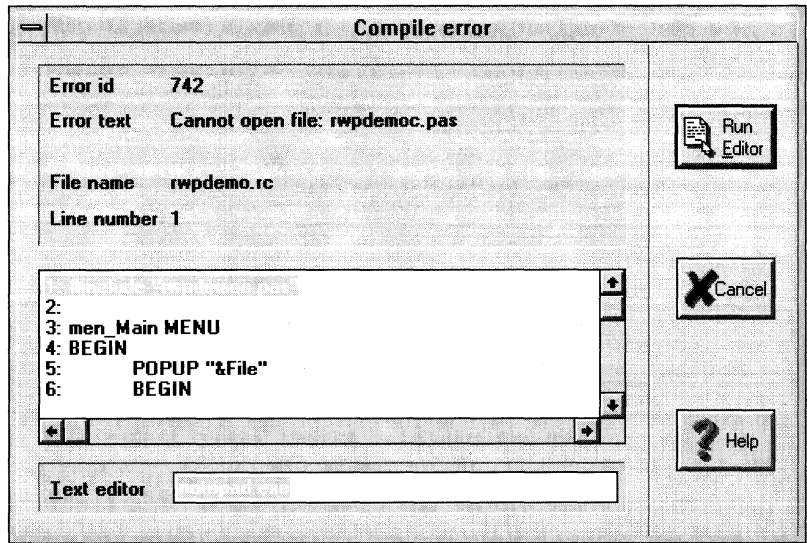
- Choose a file from the Files list. For example, if you want to open C:\TEST\MYPROJ.RC, select .RC under File Type, select the appropriate folder icons under Directories until the TEST directory is displayed, then choose MYPROJ.RC.
3. What Resource Workshop does next depends on whether the project is a binary file or a file containing resource data.
- If the project is a binary file (an executable file, a .RES file, or a DLL file), Resource Workshop decompiles the resources and shows you its progress on the left side of the status line at the bottom of the display.
 - If the project consists of a main .RC file and other files containing resource data (as is usually the case) or a single resource file containing resource data, Resource Workshop reads the project file to determine all the files in the project. Resource Workshop then works its way through all files, following references to any additional files, and compiles each resource, showing you its progress in the Compile Status dialog box.

Figure 3.6
Compile Status dialog box
for rwpdemo.rc



You can press the Cancel button to cancel the compilation. If the compiler encounters an error, Resource Workshop displays the Compiler Error dialog box, which shows you the error and highlights the line where the error occurred.

Figure 3.7
Compile Error dialog box for
rwpdemo.rc



See page 56 for more
information on setting the
Text Editor Preferences
option.

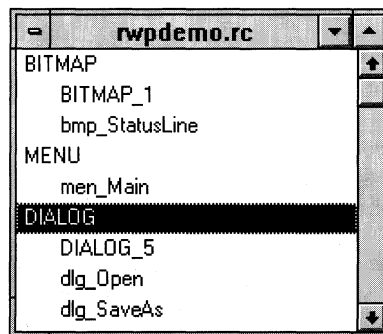
To edit the file using the external text editor specified in Preferences, press the Run Editor button. You can also specify another editor in the Text Editor text box.

When the editor comes up, make your changes, then save your changes and exit the editor. You must then reload the project.

4. Once the project is compiled or decompiled, Resource Workshop displays the Project window with all the resources listed in it.

If you open RWPDEMO.RC, you see the following Project window:

Figure 3.8
The Project window



Using the Project window

Once you've opened a new or existing project, Resource Workshop displays the Project window.

For a new project, the window is empty, and you have to put resources into it by creating them or adding them as files (more on these subjects later).

See Figure 3.8 for an example of a Project window.

For an existing project, you can see

- The complete list of files in the project
- The types of resources contained in each file
- If the file contains resource data (is not an .EXE, .RES, or .DLL file), the identifiers (**#defines** or constants) associated with the resources

The Project window acts as a file management tool, making it easy to get an overall view of a project. Even if the project contains a large number of resources, you can quickly scan through them by scrolling through the Project window.

Displaying information in the Project window

Use the View menu to determine how the Project window displays information.

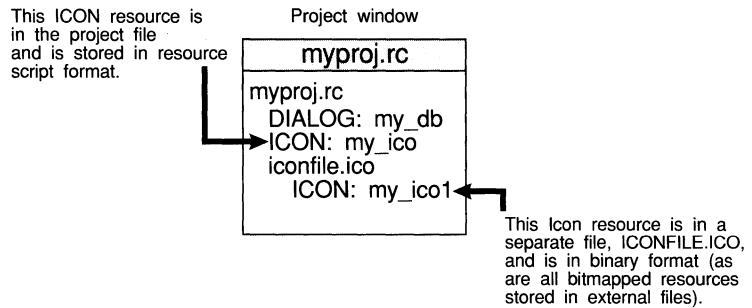
Figure 3.9
The View menu

By file
✓ By type
Show identifiers
✓ Show resources
Show items
✓ Show unused types

By File Groups resources according to the files they're in.

Choose By File to see all the file names contained in your project. In the Project window, all resources and file names are listed in the order in which they appear in the source files. Resource Workshop indicates in the Project window whether resources are saved in the Project file or in an external file, as shown in the following figure:

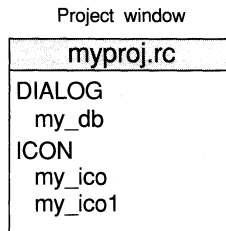
Figure 3.10
Project window showing
resources by file



By Type Groups resources according to type, such as icon, menu, dialog, identifier, and so on.

Choose By Type (and Show Resources, described shortly) if you want to see all the resources listed according to type instead of file name. In the Project window, all resources in the project are listed in this order: bitmap, menu, dialog, string table, accelerators, rcdata, cursor, icon, and any user-defined resource types you may have created. After the list of resources, any identifiers in the Project are listed if you've set Show Identifiers in the View menu.

Figure 3.11
Project window showing
resources by type



Show Identifiers Displays any identifiers (**#defines** or constants) in the project. If you don't want to see identifiers displayed in the Project window, turn this command off.

Show Resources Lists the individual names of resources, such as Cursor_1, MY_Icon, PB_BITMAP, and so on.

With Show Resources off, none of the resources defined in the project file can be selected for editing.

Uncheck this option if you want to see file names only, without a list of resources contained in those files (By File must be checked), or if you're just interested in looking at identifiers (Show Identifiers must be checked) and you want to simplify the display.

Show Items Shows another level of detail in the Project window. When Show Items is on, Resource Workshop displays items within individual resources (for example, POPUPs and MENUITEMs defined in a menu resource).

Show Unused Types Displays all possible types of resources, even if some of them might not be included in the project.



For this option to be useful, By Type must be checked.

Selecting a resource

To select a resource, use the mouse or the arrow keys to highlight it in the Project window.

- If you've chosen View | By File, look for the resource under its file name, if you know it. The resource name is preceded by the resource type and a colon.

For example, in the MYPROJ Project window in Figure 3.10 on page 39, the Icon resource my_ico1 is listed under the file iconfile.ico as ICON:my_ico1.

- If you've chosen View | By Type, look for the resource type first. The resource is listed by name under the resource type.

For example, in the MYPROJ Project window in Figure 3.11 on page 39, the Icon resource my_ico1 is listed under the ICON resource type.

Working with resources

If you're displaying resources in the Project window, you'll see them listed by name. The sections that follow tell you how to load and work with resources.

Editing a resource

When you open a resource, Resource Workshop also opens an editor for the resource. You have a choice of two types of editors:

- A visually oriented resource editor
- A text editor that lets you edit only the resource script for the resource

To pick an editor, you select the resource and choose Resource | Edit or Resource | Edit As Text. The Edit command selects a resource editor corresponding to the resource, and the Edit As Text command selects the internal text editor. Because Resource Workshop uses resource editors by default, you can load a resource and edit it by simply double-clicking the resource name in the Project window.

Using a resource editor

Each resource editor is explained in the chapter on that resource.

When you double-click a resource in the Project window, Resource Workshop loads the appropriate resource editor for the resource you choose.

- If the resource is a dialog, menu, accelerator, or string table, Resource Workshop loads an editor specially designed to work with that type of resource. For descriptions of these editors, see Chapters 4 through 7.
- If the resource is a bitmapped resource (icon, bitmap, cursor, or font), Resource Workshop displays the resource in the Paint editor. All common features of the Paint editor are described in Chapter 8. The particular Paint editor features for each resource are explained in the chapter for that resource.

Using the internal text editor

You can edit the resource script of any resource, including resources that are normally in binary format. Resource Workshop, using its internal editor, displays the text in an edit window. If necessary, Resource Workshop decompiles the binary format to let you work with the resource script.

To edit the resource script of a resource,

1. Select the resource in the Project window.
2. Choose Resource | Edit As Text. Resource Workshop opens its internal text editor and displays the script for the resource.

See the online Help index for descriptions of the script commands you can include in the resource script of a resource.



The internal text editor is similar to the Windows Notepad editor. It uses the *Del*, *Home*, *End*, *PgUp*, *PgDn*, *Tab*, and *BkSp* keys as you would expect and is always in insert mode.

When you enter text, don't spend any time formatting it, because Resource Workshop is likely to rearrange the text for you when it compiles it.

When you're finished making changes, close the text editor. Resource Workshop then asks you if you want to compile the file.

|||➡ You must click *OK* to save your changes. Clicking *No* throws away any changes you've made.

If Resource Workshop finds any errors, it tells you what they are and puts you back in the text editor so you can correct them.

If you want to edit the resource script directly without the assistance of Resource Workshop, you can open the source file with an editor of your choice and edit that file. However, note the following:

|||➡ When Resource Workshop loads the resource, it recompiles the file, reformats the resource script, and *throws away any comments in the original resource script*.

Adding a resource

You can add a resource to a project as a file reference, or you can add it directly to the project.

Adding a resource stored in a file

You can add a resource stored in an external file to the current project by using the **File | Add To Project** command. You can use this command to add a currently existing resource file or to create a new file for a new resource.

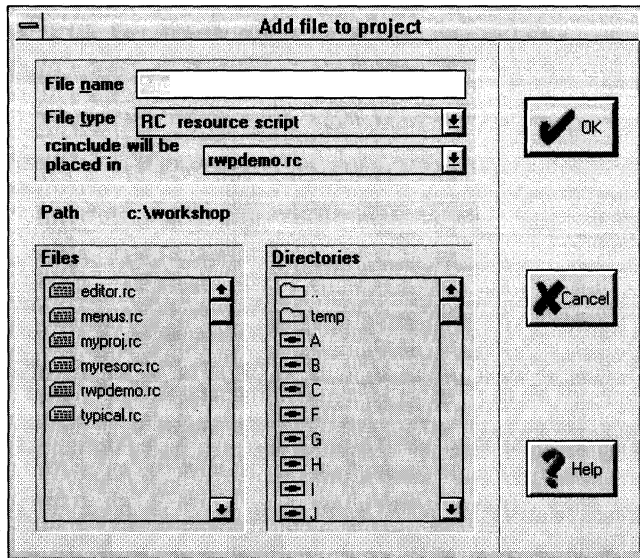
Existing resource files

If an existing resource is in its own separate file, you can add it to a project. For example, you might have a cursor you created in another project stored in a .CUR file, and you want to use it in the current project.

With the project open,

1. Choose **File | Add To Project**. Resource Workshop displays the **Add File to Project** dialog box.

Figure 3.12
The Add File to Project dialog
box



*You can also enter file search
criteria by selecting the file
type from the File Type list
box*

2. In the File Name text box, either type the name of the file containing the resource or double-click the file name if it's listed in the Files box.

If the file isn't in the current directory, you can type the file's path name or change directories by using the Directories list box. You can then search for the file by entering your search criteria in the File Name box and pressing *Enter* (for example, use *.CUR to find all cursor files). When you find the file, you can double-click it to enter it in the File Name text box.

3. In the second drop-down list (under the File Type drop-down list), you see the current project file listed, which is most likely where you will put the reference to the new file. If your project contains more than one .RC file and you want to put the reference elsewhere, scroll down the list to find the name of the file in which you want to place the reference.
4. Press *Enter* or click OK to add the file to the project. Resource Workshop puts an entry for the resource in the Project window that points to this file.

If you choose View | By File, you'll see the file name listed and under it the resource name. Any changes you make in the project to this resource are reflected in the original resource file.

New resource files

Adding a new file to a project works the same as adding an existing file (see the previous section), except that you enter the name of a nonexistent file. When you press *Enter* or click OK to add the file to the project, Resource Workshop tells you that the file doesn't exist and asks you if you want to create it. If you do, press *Enter* or click OK, and Resource Workshop creates a file of the appropriate type (based on the file extension or, if the extension isn't standard, the file type) and inserts a reference to the file in the Project window.



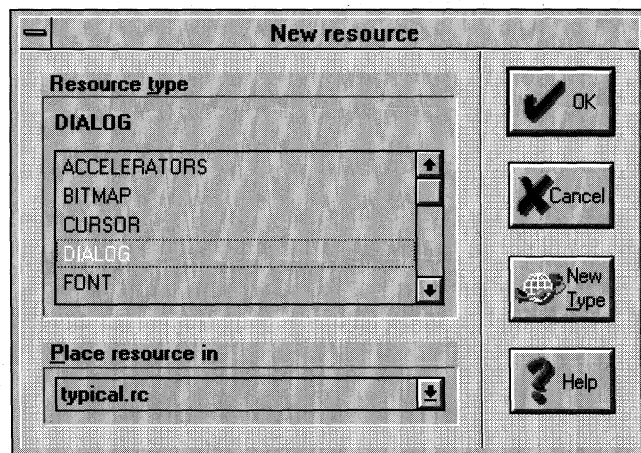
You can use this option to create a new identifier file. Just select the file type you want (H, PAS, or INC), enter the new name with the correct extension, and Resource Workshop creates the new file for you. Using this option, you never have to create identifier files using an editor outside Resource Workshop.

Creating a new resource

To create a new resource in a project as a resource script, use the Resource | New command. Make sure the project you want to work with is open, and then

1. Choose Resource | New to display the New Resource dialog box.

Figure 3.13
The New Resource dialog box



2. Double-click the type of resource you want to create.

The next steps depend on the type of resource you are creating.

See the sections on creating new resources in Chapters 9 through 12 for more information on creating new resources in binary format.

- If you create an accelerator, a menu, a dialog, a string table, an rcdata resource, or a user-defined resource, Resource Workshop puts an entry for the resource in the Project window and opens the appropriate resource editor with the new resource loaded.

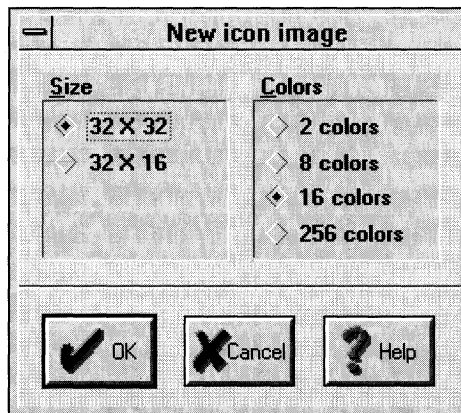
- If you create a bitmapped resource (an icon, a cursor, a bitmap, or a font), Resource Workshop asks you how you want to save the resource, as source text (a resource script in the project file) or in binary format (in an external file).

If you choose to save the resource as binary data, you need to specify the file in which to store the binary data. Resource Workshop displays the Add File To Project dialog box to make it easy to specify the file name.

- If the type of resource you want to create isn't listed, you can press the New Type button and create a user-defined type for your resource. See Chapter 13 for information on user-defined resources.

- If you create an Icon or a Bitmap resource, you see an additional dialog box that lets you specify characteristics of the resource. For example, if you're creating a new icon, you see the following dialog box:

Figure 3.14
New Icon Image dialog box

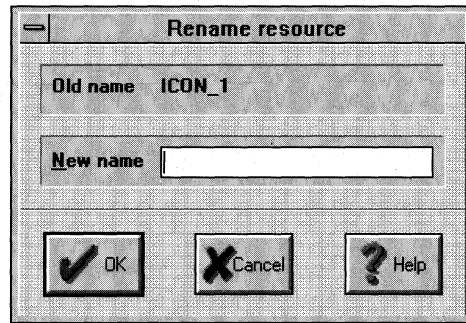


Renaming a resource

You can rename a resource as follows:

1. Choose Resource | Rename. Resource Workshop displays the Rename Resource dialog box.

Figure 3.15
The Rename Resource dialog
box



See page 49 for more
information on identifiers.

2. In the New Name text entry box, type the new resource name, then press *Enter*.
3. Resource Workshop asks you if you want to create a new identifier by that name.
 - Answer No if you want your resource to be a named type. Using a named type is slower, but it's easier to use in your code, because the Windows parameters expect a far pointer to a **char**.
 - Answer Yes if you want to assign a resource ID to the resource.



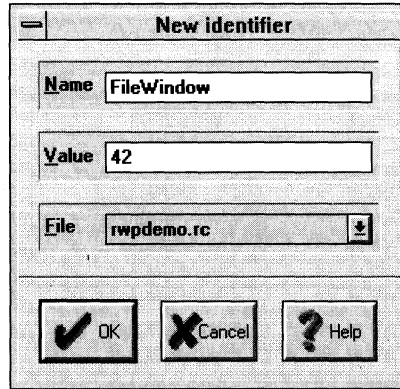
Assigning a resource ID to a resource name speeds up calls to the resource, but you won't be able to use the short integer value of the resource ID directly as a parameter. You must either typecast the integer into a long pointer to **char** or use a macro to do the typecasting for you.

If you use a macro and you write your program in C, you can use the `MAKEINTRESOURCE` macro. If your program is in Turbo Pascal, you can use the `MakeIntResource` type (a long pointer to *char*).

The `MAKEINTRESOURCE` macro looks like a function call but actually does a typecast on the identifier. For example, if you have a resource called `TEXT1` that you've assigned a value of 1001, Resource Workshop creates an identifier called `TEXT1` with that value. To use `TEXT1` as a parameter in a C program, enter it as the following expression: `MAKEINTRESOURCE(TEXT1)`.

4. If you answer Yes (you do want to create a new identifier with the same name as the resource), Resource Workshop displays the New Identifier dialog box.

Figure 3.16
The New Identifier dialog box



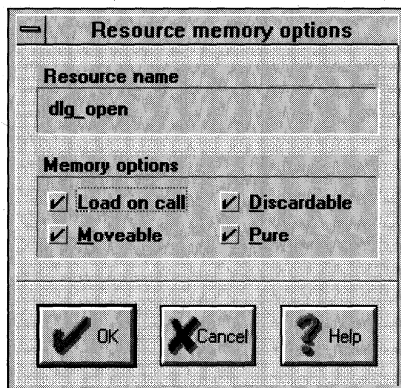
5. Type an appropriate integer value in the line provided and make sure you select the file in which you want to store the identifier. Use an integer greater than 256. Resource IDs from 1 to 256 are reserved for predefined resource types and future expansion.

Specifying resource memory options

Resource Workshop makes it easy to specify how each resource in your project should be managed in memory. However, it's probably better to leave the memory settings at their defaults unless you're an experienced Windows programmer; you might not be able to foresee the implications of changing the way a resource is handled in memory.

To specify memory options, click the resource in the Project window to select it. Then choose Resource | Memory Options. The Resource Memory Options dialog box appears.

Figure 3.17
The Resource Memory
Options dialog box



Uncheck any load or memory options you don't want. For some bitmapped resources, you might want to uncheck the Discardable option. If this option is unchecked, the application can modify the resource while it's in memory.



If you set memory options for an Icon resource, those options apply to all the images in that resource.

Here's what each option in the Resource Memory Options dialog box does:

Table 3.1
Resource Memory Options

Option	Description
Load on Call	Loads the resource into memory only when it's needed. Choosing Load On Call can reduce the amount of time required to load your program. If you uncheck this option, you'll activate Preload, which means that Windows will load the resource into memory when it first loads the program. You need to preload a resource only if you know Windows needs the resource as soon as the application begins to execute.
Moveable	Lets Windows move the resource segment in memory to make room for other memory allocations. If you uncheck this option, the resource segment occupies a fixed block of memory.
Discardable	Lets Windows discard the resource segment from memory when it's no longer needed. Windows can load the resource into memory again when necessary.

Table 3.1: Resource Memory Options (continued)

Option	Description
	If you uncheck this option, you'll activate Non-discardable. Windows won't be able to remove the cursor segment from memory while the application is running, and, if Pure isn't checked, you'll be able to modify the resource from within your application.
Pure	Prevents the resource segment in memory from being modified. Usually, you'll want to leave this option checked. See the Windows documentation for information about this option.

Removing a resource

To delete a resource from a project, select the resource in the Project window, then choose either Edit | Cut or Edit | Delete to remove it. (Edit | Cut let you paste the resource elsewhere.)

Using identifiers

See page 22 for a description of the different types of identifier files.

Windows requires that every resource have a unique name or a unique integer (called a *resource ID*) associated with it. You can use Resource Workshop to assign actual integers as resource IDs, but it's more likely that you'll want to use an identifier to represent the resource ID. The term *identifier* refers to a C **#define** or a Pascal constant used to assign to a resource a name that's more meaningful than an integer.



See "Working with binary files" on page 58.

If you're working with a .RES file, an executable file, or a DLL, Resource Workshop decompiles all resource IDs in these files into integer values. You can't use identifiers as resource IDs in these types of files, but you can save the file as an .RC file and then assign identifiers to your resources.

For applications written in C, it's common to use **#defines** to assign names to resources. If you use **#defines**, you can store them in one or more header files.

For applications written in Pascal, you use constants for this purpose. If you use constants, you can store them in one or more units or include files.

- If you use a Pascal unit or include file with Resource Workshop, you can assign only numeric values to the constants in the file.
- To simplify things, header files, units, and include files are called *identifier files* in this book.

Adding an identifier file

You can't add an identifier file to a decompiled object file, such as an executable file or a DLL.

If you don't already have a header file associated with your C source code, or a unit or include file associated with your Pascal program, you can create a new identifier file with Resource Workshop.

To add an identifier file to your project, choose File | Add To Project and use the Add File To Project dialog box. See page 43 for information on using this dialog box.

Working without an identifier file

If you don't add an identifier file to your project, Resource Workshop can store any identifiers you create for a resource in the active project file. It stores these identifiers as **#defines**, not as Pascal constants.

However, if you decide later to pull out the **#defines** and put them in one or more header files, you'll have to use a text editor to cut them from the resource scripts and paste them into the header files. Because this process can be time consuming, is likely to cause errors, and supports only **#defines**, it's better to set up your identifier files first, before you start working on your resources. Then, as you create your resources, you can put each identifier in the appropriate identifier file, because Resource Workshop asks you where to store an identifier when you first create it.

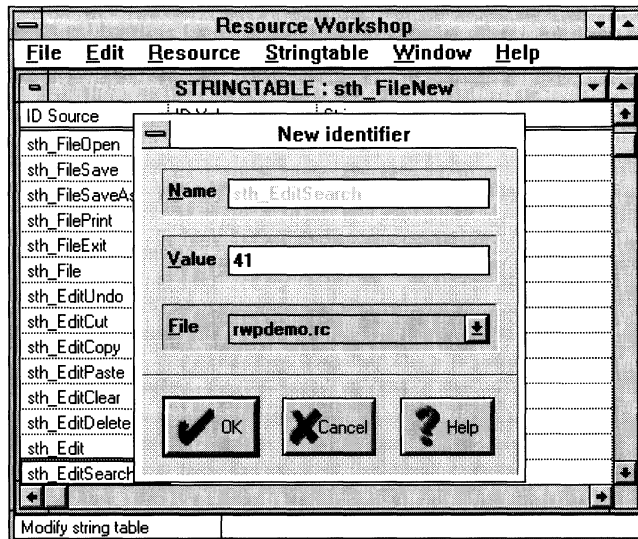
Adding identifiers by using a resource editor

See Chapter 7 for a description of the String editor.

Most often, you'll add an identifier to a file by creating a new identifier in a resource editor.

For example, let's say you're editing a string table. Every string in a string table requires a unique identifier, which you enter in the ID Source field. If you type an identifier name that the String editor doesn't recognize and press *Enter*, Resource Workshop asks you if you want to create a new identifier. If you click Yes, you'll see the New Identifier dialog box.

Figure 3.18
The New Identifier dialog box
on the String Table editor



In the Value box, you can either enter a new value for the identifier or accept the displayed value, which is guaranteed to be unique for the string table.

In the File drop-down list, you can specify the file in which to store the identifier. Since you should already have created at least one identifier file for the current project, you can scroll to that file, select it, and then either click OK or press *Enter* to put the identifier in that file.

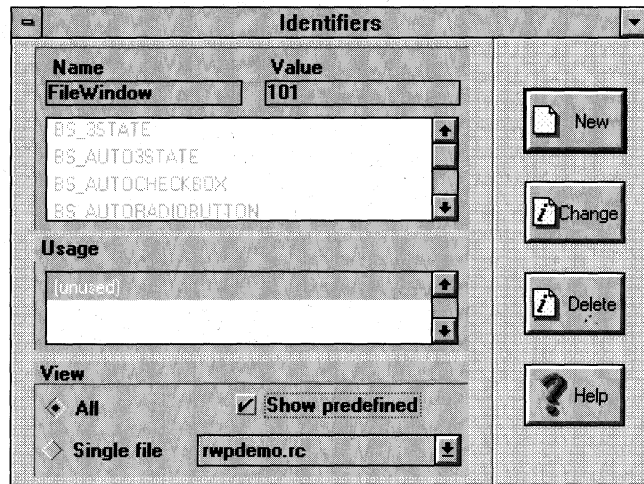
You can also add a new identifier to a file directly, without first associating the identifier with a resource. The next section tells how to work directly with identifiers.

Adding, editing, deleting, and listing identifiers

To add, edit, delete, or just list identifiers,

1. Choose Resource | Identifiers. Resource Workshop displays the Identifiers dialog box.

Figure 3.19
The Identifiers dialog box



This dialog box is a *modeless* dialog box. You don't have to exit it in order to perform other functions; instead, you can keep it around just as you would one of the editor windows. If you do want to get rid of it, double-click the Control-menu box in the top left corner of the dialog box.

2. Use the radio buttons and the check box under View to control which identifiers Resource Workshop displays in this dialog box:
 - The radio buttons are All and Single File.
 - Choose All if you want to display identifiers you have added to your project.
 - Choose Single File if you want to display identifiers only for one of the files in the list box to the right of this choice. If you choose Single File and there's more than one file in the list, be sure to highlight the file you want.
 - Check Show Predefined if you want to display the predefined **#defines** that are included with Windows.
3. The identifiers are displayed in the top list box. The Name and Value of the selected identifier are displayed under those headings, and the files in which the identifier is used are displayed in the list box under Usage.
4. Once you have displayed the identifiers you want to work with, you can use the buttons on the right side of the dialog

box to add, change, or delete identifiers. Here's a little more information about these buttons:

- Clicking the New button displays the New Identifier dialog box. See Figure 3.18 for an example of this dialog box.
If your project contains more than one unit, include file, or header file, make sure you use the drop-down list next to File to choose the file where you want to insert the identifier.
An identifier must be unique. Only the first 31 characters are significant: Resource Workshop ignores any characters past the 31st character.
- If you want to change or delete an existing identifier, first select the identifier from the top list box, then click either the Change or the Delete button.



Predefined identifiers are grayed out and can't be changed or deleted.

Saving resources, files, and projects

It's a good idea to save your work often. Resource Workshop provides you with a variety of save commands so you can choose exactly what you want to save and how to save it.

File | Save Project Choose File | Save Project to save everything in your current project. If you're saving a new project that hasn't been named yet, Resource Workshop displays a dialog box that lets you specify a name and directory.

Resource Workshop always saves the project file and any files it references. If you have selected .RES or .EXE from the File | Preferences dialog (described on page 55), Resource Workshop also compiles and saves to a .RES file or binds the resources to an executable file or DLL.

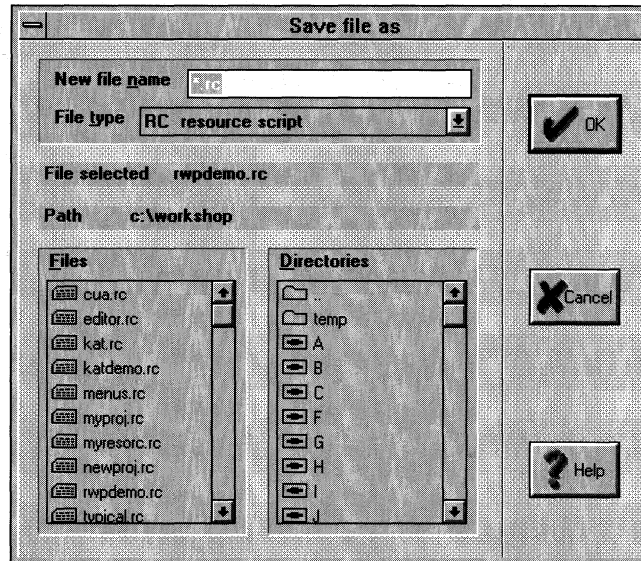
If your project is based on an .RC file, Resource Workshop also compiles the project as part of the save process. This compiled version is stored in a file with an .RWS extension. The next time you open this project, Resource Workshop can save time by loading the already compiled .RWS file instead of having to load and compile the .RC file.

Resource Workshop is capable of doing any compiling you might require for your resources. See Figure 2.11 on page 26 for an example of how a project might be compiled.

File | Save File As

If you want to rename the current project or resource file, choose File | Save File As. Resource Workshop displays the Save File As dialog box.

Figure 3.20
The Save File As dialog box



Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or include the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or click OK to save the file.

Resource | Save Resource As

To put a resource in a separate file for use with other projects,

1. Choose Resource | Save Resource As. Resource Workshop displays the Save Resource As dialog box.

This dialog box shows almost the same information as the File Save As dialog box (see Figure 3.20). The primary difference is that, on the third line, instead of showing you the file it's selected to save, it shows you the resource you'll be saving.

2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or include the path when you type the file

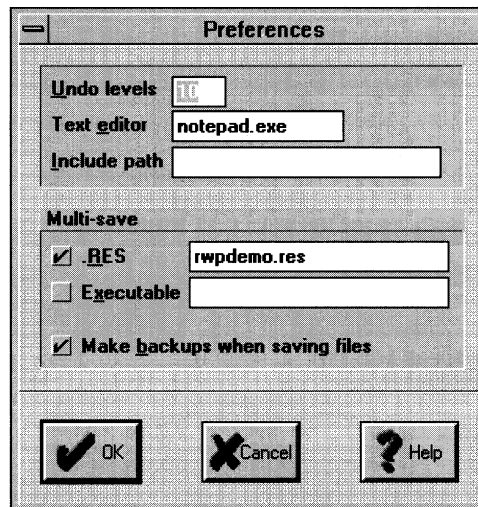
name. When you're satisfied that the file name is correct, press *Enter* or click OK to save the file.

3. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on.
 - Clicking Yes causes the resource script or current file reference in the Project file to be replaced by a reference to the new file. All future changes to the resource will be saved to the external resource file and not in the project file or in any previous resource file.
 - Clicking No just creates the resource file without making any changes to the Project file.

Configuration preferences

To set configuration preferences, choose **File | Preferences**. Resource Workshop displays the Preferences dialog box.

Figure 3.21
The Preferences dialog box



In this dialog box, you can make the following choices:

- Undo Levels** Resource Workshop has a multilevel Undo and Redo feature that lets you correct actions in any of the resource editors. Depending on the amount of memory in your computer, you can undo or redo up to 99 actions. The default number of levels is ten.

For example, if you're working in the Paint editor on an icon and you fill an area with a color, then select and move a portion of the image, only to find that the fill now doesn't look right, you can undo the move, then undo the fill (press *Alt+BkSp* or choose Edit | Undo for each undo). If you want to see how the fill looks again, you can redo the fill, and then, if you like, you can redo the move again (press *Shift+Alt+BkSp* or choose Edit | Redo for each redo). You can work your way back and forth through your edit session this way through as many levels as you have set in the Undo Levels option.

Text Editor When Resource Workshop loads in a project file, it compiles all the resources in the file. If the compiler encounters an error, it stops compiling, notifies you that there was an error, and asks you if you want to edit the file by using the external text editor you have specified in this option.

The default external text editor is the Windows Notepad editor. If you specify another editor, it must be one that runs under Windows. (A DOS editor with a PIF file will work.)

Include path This is the path Resource Workshop will search for files containing identifiers (C header files, Pascal units, or include files). You can set this option only if you do not have a project open. When you choose this option, Resource Workshop saves it in WIN.INI as the default include path.

Multi-Save The .RES and Executable preferences control how a project is to be saved when you select File | Save Project. These preferences are enabled only when a resource compiler (.RC or .DLG) project is open because they apply only to a specific project. Regardless of the multi-save settings, the project always gets saved in its original format as well. (For example, if the project is an .RC file, the resources in the file are always saved as resource scripts in addition to any multi-save options.)

.RES

Compiles the current project's resources and saves them in .RES format (in binary format).

Executable

Compiles the current project's resources and binds them to the executable file specified in this option (can be an .EXE or .DLL file).

Make backups when saving files

If you check the Backups option, Resource Workshop creates an additional set of backup files each time you save a project. Backup files have a tilde (~) as the first character in the file extension. For example, when you save MYPROJ.BMP, the backup file is called MYPROJ.~BM.

Copying resources between projects

There are two ways you can copy a resource from the current project to another project:

- One way is to save the resource as a file, close the current project, open the other project, and add the resource as a file to the new project. If there are any identifiers in the resource, you have to be careful they're preserved when you add the resource to the new project.
- An easier way is to have two copies of Resource Workshop open, one for each project, and to use the Windows clipboard to copy the resource from one project and paste it to the next. This method is not only faster than the first method, but it also saves all the identifiers.

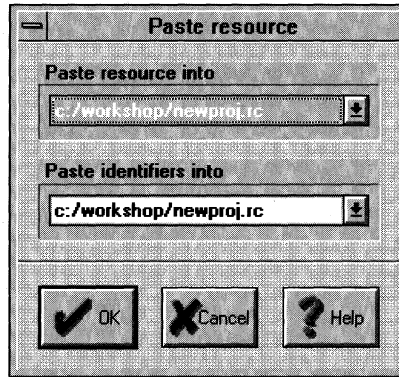
To copy a resource using the second method,

1. Open two copies of Resource Workshop, one with the project containing the resource you want to copy (the source project) and another with the project you want to copy the resource to (the target project).
2. Be sure the target project has a reference to an identifier file that will receive any identifiers in the new resource. (If necessary, choose File | Add To Project and add the appropriate type of identifier file.)
3. Select the source project, then select the resource you want to copy in the Project window. Choose Edit | Copy to copy it to the Windows clipboard.

*Don't use the Ctrl+Ins hotkey—
it won't copy the resource
properly.*

4. Select the target project, then, with the Project window active, choose Edit | Paste to paste the resource into the project. Resource Workshop displays the Paste Resource dialog box.

Figure 3.22
The Paste Resource dialog
box



5. The Paste Resource Into text box should contain the name of the target project. Make sure the Paste Identifiers Into text box contains the name of the identifier file that will receive any identifiers in the resource (if necessary, scroll the drop-down list and choose the correct file name), then press *Enter* or click OK to paste in the new resource.

Working with binary files

You can customize the user interface of an executable file. For example, you might want to translate it into another language.

Resource Workshop allows you to load executable files, .RES files, and DLLs as projects. What Resource Workshop does when you load one of these types of files is to decompile the resources in the file and show them to you as though they were part of a regular .RC file. When you're finished with your changes, Resource Workshop compiles the resources again into binary code and stores them in the original file.

Because the resources you're working with in this type of project are never stored as resource scripts, you can't assign any identifiers to the resource IDs. (In other words, all resource IDs must be integers in binary files.) However, what you *can* do is save the project as an .RC file. Once you do so, the resources can be saved as resource scripts, so it's OK to assign identifiers to them.

There's another reason you might want to save the resources as an .RC file. If you're customizing the user interface of a program and you have access only to the executable file or DLL, you'll want to save your changes in a separate .RC file so you can apply the changes to the next version of the program. Of course, you'll have to make sure that the resources in your .RC file have the same resource IDs as their counterparts in the new version and that they are otherwise compatible with the new version.

Once you save the project as an .RC file, Resource Workshop won't automatically save the resources back to the original file. You have to enter the original file name as a multi-save option in the Preferences dialog in order to save the resources there as well.

The whole process works as follows:

1. Choose File | Save File As and use the File Save As dialog box to enter the name of the new .RC file. When you press *Enter* or click OK to save the file, you'll see the name of the project change to the new file name.
2. Choose File | Preferences and enter the name of the original binary file as a Multi-save option.
 - If the original binary file was a .RES file, check .RES and enter the name in that text box.
 - If the original binary file was an executable or DLL file, check Executable and enter the name in that text box.
3. Choose File | Add To Project and specify an identifier file to hold the new identifiers. If the file you specify doesn't exist, Resource Workshop creates a new one for you.
4. Make your changes to the resources and specify identifiers where you want them. For each new identifier, Resource Workshop asks you if you want to save it in the identifier file.
5. When you quit and save the file, Resource Workshop saves both the .RC file and the binary file. If the binary file is an executable file or a DLL, the changed resources are bound into it and are available immediately when you run that program.

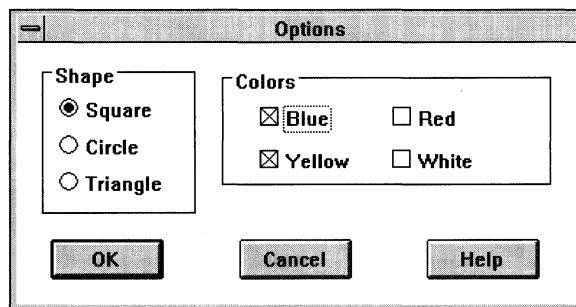
Be sure to preserve the current integer values of the resource IDs.

Creating dialog boxes

Dialog boxes give the user a way to interact with your application. A dialog box is usually a pop-up window that lets the user specify information (files to open, colors to display, text to search for, and so on).

When a dialog box is displayed in an application, it's shown as a window. The dialog box usually contains a number of *controls*, such as buttons, text boxes, and scroll bars. Controls usually let the user specify information, but can also be used to display static text and graphics in a dialog box.

Figure 4.1
A typical dialog box



From a user's perspective, controls are often the easiest way to make choices. For example, by clicking buttons, checking boxes, or selecting items from a list, the user can turn options on and off or decide what the program should do next.

From the programmer's perspective, the dialog box is the *parent window* and each control is a *child window* acting as an input

device. To create a dialog box, you fill an empty dialog box with the controls you want. Each control is responsible for processing mouse and keyboard messages, based on what the user types or clicks.

Resource Workshop's Dialog editor makes it easy to create and edit dialog boxes for your application. When working with dialog boxes, you perform four primary tasks. You

- Start the Dialog editor
- Customize a dialog box
- Test the dialog box
- Save the dialog box

The first task, starting the Dialog editor, displays a dialog box ready for you to customize. The second and third tasks, customizing the dialog box and testing it, are functions of the Dialog editor itself. The fourth task saves the dialog box you created or edited.

At the end of this chapter is a sample project that illustrates how to use the Dialog editor. See page 107 to work with the sample project.

Starting the Dialog editor

Whether you want to create a new dialog box or edit an existing one, you'll use the Dialog editor. When the Dialog editor starts, it displays a dialog box you can edit to make it fit the needs of your application. How you start the Dialog editor depends on whether you want to create a new dialog box or edit an existing one.

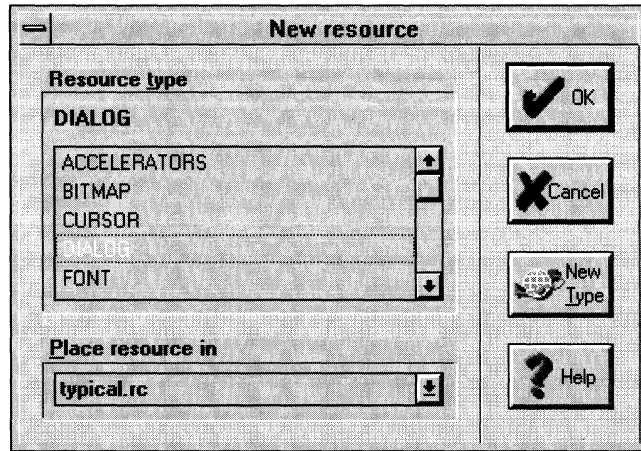
Creating a new dialog box

See Chapter 3 if you need information about opening a project.

To start the Dialog editor to create a new dialog box,

1. Open a project.
2. Choose Resource | New. The New Resource dialog box appears.

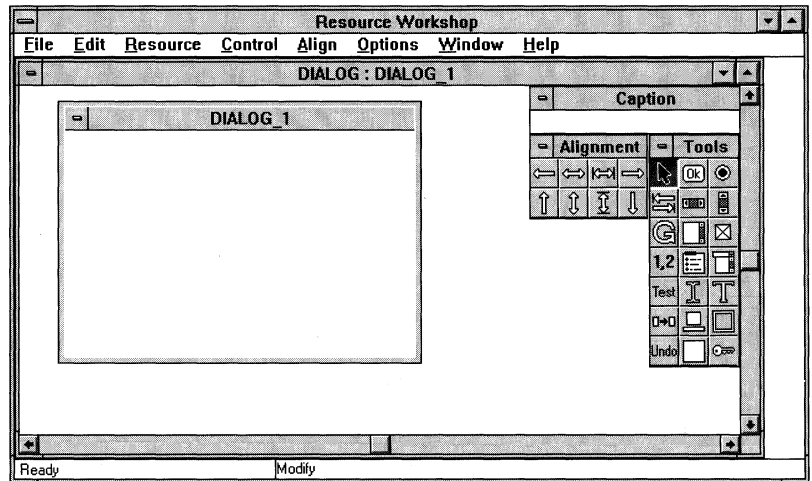
Figure 4.2
The New Resource dialog
box



3. In the Resource Type list box, select DIALOG and choose OK.

You're now in the Dialog editor. The empty dialog box you'll "fill in" is on the left side. A Caption window, an Alignment palette, and a Tools palette are on the right. At the bottom of the Dialog editor is the status line.

Figure 4.3
The Dialog editor with an
empty dialog box



Editing an existing dialog box

See Chapter 3 if you need information about opening a project.

To start the Dialog editor to edit a dialog box that already exists in a project file,

1. Open the project that contains the dialog box you want to edit.
2. Locate the dialog box you want to edit in the Project window and double-click it.

The Dialog editor opens and the dialog box you selected to edit appears on the left. The Caption window, the Alignment palette, and Tools palette are on the right, and the status line is at the bottom of the Dialog editor.

Customizing a dialog box

Once in the Dialog editor, you can define your dialog box. You can also add, change, group, reorder, move, resize, or delete dialog controls so that your dialog box functions the way you want it to.

Defining a dialog box

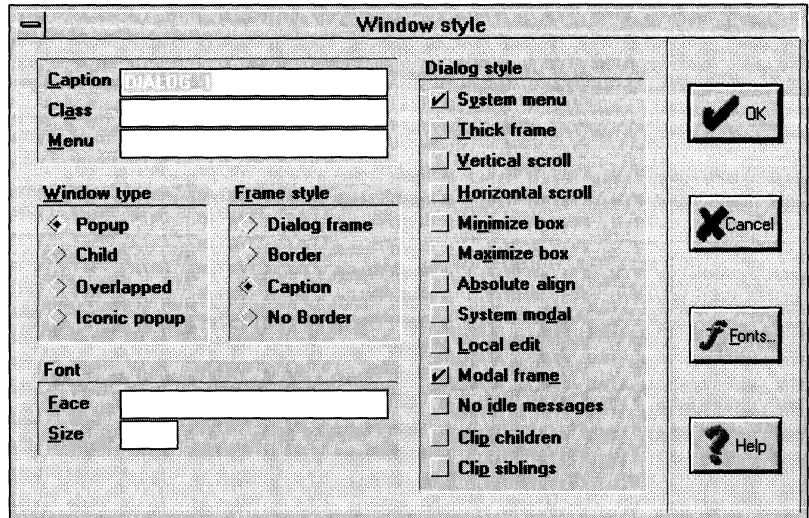
With Resource Workshop, you can specify a caption for your dialog box, decide what type of window it will be, choose which fonts will be displayed in it, and so on.

Click the title bar or outer edge of the dialog box to select it. When a dialog box is selected, Resource Workshop displays a *sizing frame* around it.

Now you can make any changes you want to the dialog box window. You can

- Resize the dialog box by dragging the mouse cursor from the appropriate edge or corner.
- Move the dialog box by dragging the mouse cursor from within the dialog box, or by using the arrow keys.
- Press *Enter* or double-click anywhere in the window to display the Window Style dialog box. (If the window isn't already selected, you must double-click its title bar or outer edge.)

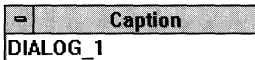
Figure 4.4
The Window Style dialog box



Use the Window Style dialog box to specify more detailed information about such things as captions, frame styles, and fonts for your dialog box.

Adding a caption

There are two ways to add a caption to your new dialog box. You can use the Caption window or the Window Style dialog box.



To add a caption with the Caption window,

1. Click the title bar of the dialog box to select it.
2. Type a new caption while the dialog box is selected and you'll see it appear in the title bar, or press *F6* to move to the Caption window and replace the name selected by Resource Workshop by typing in a new caption.
3. Press *Enter*.

To add a caption with the Window Style dialog box,

1. Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box, or by selecting the dialog box and pressing *Enter*.
2. Next to Caption, type the caption you want to appear at the top of your dialog box.
3. Under Frame Style, choose Caption. The caption won't be displayed if you choose any other frame style.
4. Choose *OK*.

Choosing the window type

Open the Window Style dialog box to choose a type for your dialog box. This table lists your choices:

Table 4.1
Window types

Type	Description
Popup	A pop-up window. Because most dialog boxes are popups, this is the default.
Child	A child of the current window.
Overlapped	A overlapped pop-up window that can be covered or partially covered by another. You'll want to define a dialog box as overlapped only when it's the main window in the application.
Iconic Popup	A pop-up window that is originally displayed as an icon.

Choosing a frame style

The frame style of the dialog box determines the appearance of the dialog box frame and whether the dialog box displays a title bar at the top.

You choose a frame style for your dialog box from the Window Style dialog box. Your choices are listed in this table:

Table 4.2
Frame styles

Style	Description
Dialog Frame	A double border, without a title bar.
Border	A single, thin border, without a title bar.
Caption	A single, thin border and a title bar where a caption can be displayed (default).
No Border	No border and no title bar.

Specifying the dialog style

The dialog style determines what the dialog box looks like and how the user can work with it. You can choose one or more of the following styles for your dialog box:

Table 4.3
Dialog box styles

Style	Description
System Menu	Includes a System menu box on the left side of the title bar. A System menu is also called a Control menu. The System menu appears only if you've also chosen Caption for the dialog box frame style. (Caption and System Menu are both defaults.) If the dialog box is defined as a child window, you'll get a Close box instead of a Control menu.
Thick Frame	Places a thick frame around the dialog box. This option defines what the user will see when the dialog box appears within an application. If you want the dialog box to be resizable, use this option. (Don't confuse this option with the Thick Frame option in the Dialog editor Preferences command. That option defines what the dialog box looks like when you select it in the Dialog editor.)
Vertical Scroll	Adds a vertical scroll bar to the dialog box frame.
Horizontal Scroll	Adds a horizontal scroll bar to the dialog box frame.
Minimize Box	Adds a Minimize button on the right side of the title bar. The Minimize button appears only if you've also chosen Caption for the dialog box frame style.
Maximize Box	Adds a Maximize button on the right side of the title bar. The Maximize button appears only if you've also chosen Caption for the dialog box frame style.
Absolute Align	Makes the dialog box coordinates relative to the display screen rather than the parent window.
System Modal	Makes the dialog box system modal, meaning the user can't switch to do anything else until the dialog box is put away.
Local Edit	Allocates any edit text controls included in this dialog box to the application's local heap. Choose Local Edit if your application needs to use EM_SETHANDLE and EM_GETHANDLE messages.

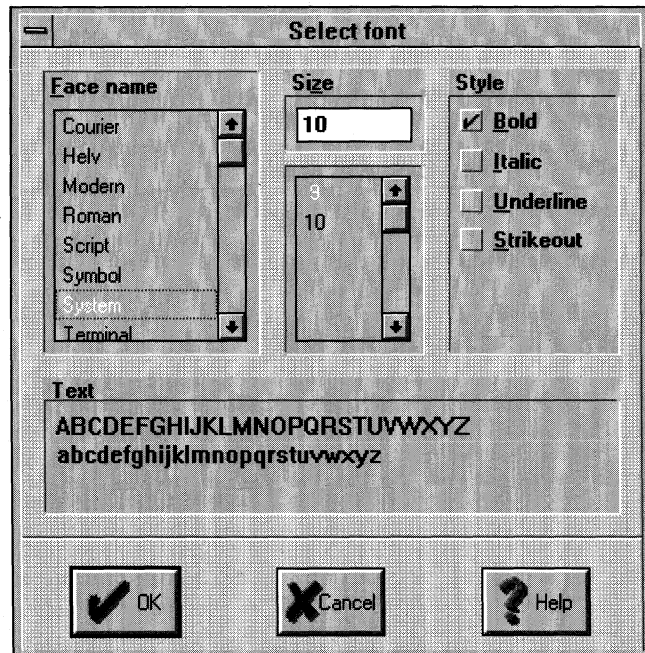
Table 4.3: Dialog box styles (continued)

Style	Description
Modal Frame	Frames the window with a combination of the dialog frame and caption styles (default). Choose Modal Frame if you want users to be able to move the dialog box.
No Idle Messages	Suppresses sending WM_ENTERIDLE messages to the application's main window. The dialog box must be modal for this option to take effect.
Clip Children	Protects the client area of child windows from being drawn on by the dialog box window.
Clip Siblings	Protects the siblings of this window. Drawing is restricted to this window. This option is not required for pop-up windows, but can be useful for child dialog windows.

Specifying fonts To choose which font the text in the dialog box uses,

1. Open the Window Style dialog box.
2. Click the Fonts button to open the Select Font dialog box.

Figure 4.5
The Select Font dialog box



3. Use the Select Font dialog box to choose a typeface, size, and style for the text in your dialog box.



Note the characters displayed at the bottom of the Select Font dialog box. They change to show you the current typeface, size, and style you've selected.

Including a menu

Because it's really a window, a dialog box could include a menu, but seldom does.

To include a menu in your dialog box,

See Chapter 5 for details about defining menus.

1. Define the menu as a separate resource and make sure it's added to the project. Be sure to note the resource name or numeric ID that identifies the menu.
2. Open the Dialog editor for the dialog box you want to add the menu to.
3. Open the Window Style dialog box.
4. In the Menu input box, type the menu's resource name or numeric ID.

The Dialog editor won't display the menu resource exactly as you've defined it. Instead, it displays a pop-up window called "Menu" that includes a single menu item called "Item."

Assigning a custom class to a dialog box

If you're an experienced Windows programmer, you might want to assign a custom class to a dialog box. Then you can process dialog box messages with your own windows procedures instead of using Windows 3.0 procedures.

To assign a custom class to a dialog box,

- Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box, or by selecting the window and pressing *Enter*.
- Type the class name in the Class input box.

If you don't understand custom classes or the CLASS statement, you probably don't need to assign the dialog box to a custom class.

Working with controls

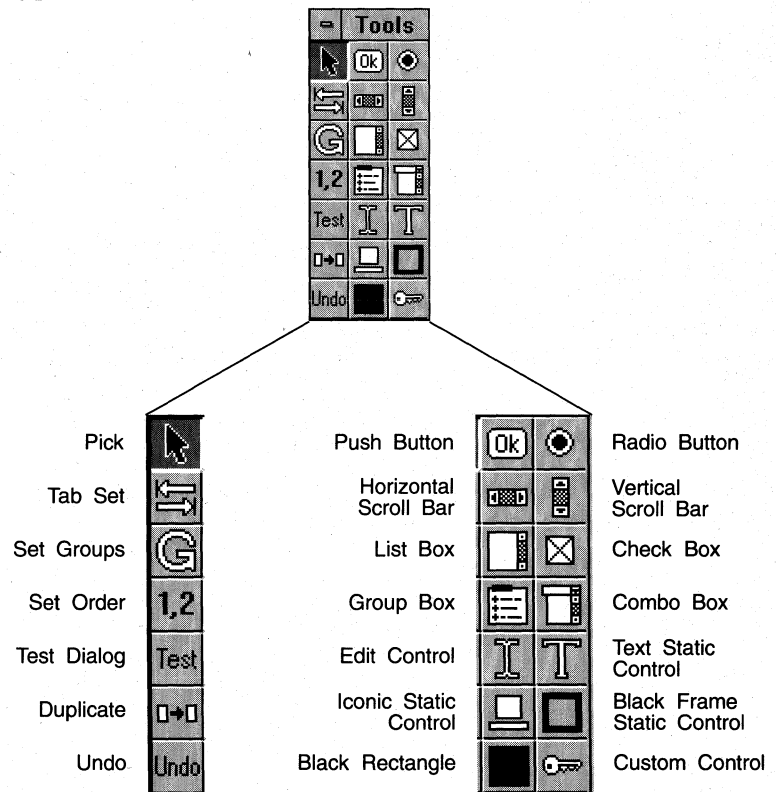
Controls let users interact with dialog box data.

In general, controls fall into these categories:

- Buttons
- Scroll bars
- List boxes
- Edit text
- Static controls (including text, icons, rectangles, and frames)
- Combo boxes
- Custom controls

In the Dialog editor, the Tools palette makes it easy to create the type of control you want.

Figure 4.1
The Tools palette



The two rightmost columns of the Tools palette are the control tools. The left column contains tools that put the Dialog editor in

various modes; you can find information about them on pages 72 to 101. Here are brief descriptions of the dialog controls you can put in a dialog box using the control tools in the Tools palette.



Push button

A rectangular button the user “presses” to select an action. Push buttons always contain text, so you will need to specify a caption for each one.



Radio button

A circular button with text on its left or right side. When the button is selected, a solid dot fills the circle.



Horizontal scroll bar

A horizontal rectangle with direction arrows on each end.



Vertical scroll bar

A vertical rectangle with direction arrows on each end.



List box

A rectangle usually containing a list of text strings. If you use the Owner-draw style, the list box can also contain a visual representation of a list of data. Usually, a user can browse through what's displayed in a list box, then select one or more items. You'll often see list boxes used in a File Open dialog box.



Check box

A rectangular button with text to the left or right. When a check box is selected, an X appears on the button. When a check box is unselected, the X disappears. Check boxes are often used to represent Boolean (on/off) data.



Group box

A rectangular box used to visually group other controls together. You can include a caption to display in the upper left corner of the group box.



Combo box

A combination of a list box and edit text control or a list box and static control.



Edit text control

A rectangle into which the user can enter text from the keyboard.



Text static control

Text that appears in the dialog box.



Iconic static control

An icon.



Black frame static control

An empty, rectangular frame that takes the color of the current window frame.



Black rectangle

A static control icon appearing as a rectangle that's the same color as the current window frame.



Custom control

A control that doesn't fit into any of the predefined Windows types and, therefore, has a different window class than the predefined types do.

Adding controls

The easiest way to add a new control to your dialog box is to click the control you want in the Tools palette. Your cursor will change to indicate the type of control you are placing. Click where you want to place the control in the dialog box.



If you select a control from the Tools palette and then change your mind about placing it, choose the Pick tool. Your cursor will return to the familiar arrow shape and you will be able to select controls in your dialog box. Pressing *Esc* also cancels the placing of a control and returns you to selection mode.

You can also use the Control menu to add controls to your dialog box:

1. Use your mouse or press *Alt+C* to open the Control menu.
2. Choose the control type you want to add to your dialog box.
3. Click in the dialog box where you want the control placed.

Adding multiple copies of a control

Once you place a control in your dialog box, you can place additional copies of the same control. Move to where you want another copy to appear and click the right mouse button.

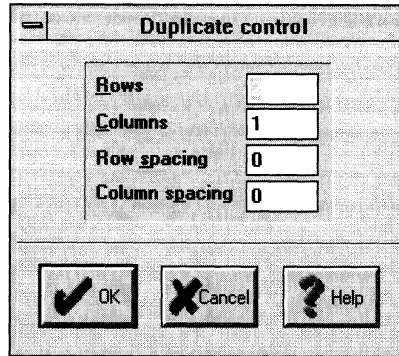
You can also place multiple, identical copies of a control in rows or columns. For example, you might want to place two columns of four check boxes in your dialog box. You could place each check box individually, but Resource Workshop gives you an easier way with the *Edit | Duplicate* command.

To place multiple copies of a control in rows or columns,

1. In your dialog box, select the control you want to duplicate.



Figure 4.7
The Duplicate Control dialog
box



2. Select the Duplicate tool or choose Edit | Duplicate. The Duplicate Control dialog box appears.

3. Specify the number of rows and columns you want, as well as the spacing in units between the rows and columns. For example, if you want eight check boxes placed in two columns, you would specify four rows and two columns.
4. Choose OK. Neatly aligned, multiple copies of your control appear in your dialog box.

Editing controls

You can modify a control you've already added to your dialog box. Just double-click the control and the Style dialog box appears. The options in this dialog box vary according to the type of control you are working with.

If you're using the keyboard, use *Tab* to select the control you want to edit. Press *Enter* to display the Style dialog box for the selected control.

Selecting multiple controls

When you are editing controls, you might want to perform an editing operation on more than one control at a time. Resource Workshop has several editing options that can be used on multiple controls. Before you can use them, however, you need to know how to select more than one control at a time.

(Your cursor is the Pick Rectangle icon when selection mode is on.)

2. Click in the dialog box and continue to hold down your left mouse button. The position you click is the starting point for the sizing frame that appears as you drag your mouse.
3. Drag your mouse so that the sizing frame surrounds all the controls you want to select.
4. Release the mouse button. A box will appear around each selected control.

If you have selected a group of controls and want to add one or more controls to the group, or want to delete one or more controls to the group, hold down *Shift* and click the controls you want to add to or delete from the group.

Moving and resizing controls

Click a control once to select it. Move the control by dragging from anywhere within it. Or move the mouse cursor to the appropriate edge or corner (until it turns into a double arrow), then drag to resize the control.

You can use the keyboard and the mouse to “fine tune” the size:

1. Move the mouse cursor over the appropriate border.
2. Hold the left mouse button down.
3. Press an arrow key once to move the mouse cursor a single dialog unit.

If you want to use the keyboard only, *Tab* selects the control. Then use the arrow keys to move what you’ve selected, and press *Enter* to confirm the move. Press *Esc* instead of *Enter* if you change your mind.

Once you’ve added a control to your dialog box, you can specify exact x- and y-coordinates for that control. The x- and y-coordinates control not only positioning, but also the size of the control.

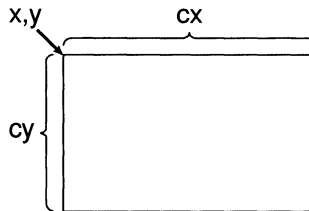
Here’s how to specify x- and y-coordinates:

1. Choose the control you want to specify coordinates for. You can click the control or use *Tab* to select it.
2. Choose *Align | Size Controls* or hold down the *Alt* key while double-clicking the mouse. The Size Controls dialog box appears.

3. Specify the x , cx , y , and cy coordinates in dialog units. In a dialog unit, y equals $\frac{1}{8}$ of the font height, and x equals $\frac{1}{4}$ of the font width.

The x - and y -coordinates define the starting point of the dialog control, the cx value defines the horizontal width of the dialog control, and the cy value defines the vertical height of the dialog control, as shown here.

Figure 4.8
Dialog box coordinates



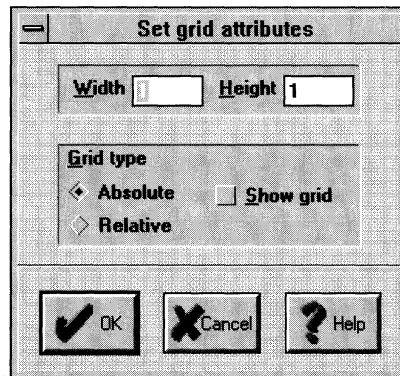
Aligning controls with a grid

When moving controls around to your satisfaction, you can display a grid on your dialog box and use it to align your controls.

To display a grid,

1. Choose **Align | Grid**. The Set Grid Attributes dialog box appears.

Figure 4.9
The Set Grid Attributes dialog box



2. Specify the desired width and height of your grid.
3. Select the Grid Type. These are your options:

Table 4.4
Grid Type options

Grid Type	Description
Absolute	Snaps the control to the nearest grid line.
Relative	Moves the control only in increments of the grid width horizontally and the grid height vertically. Therefore, if a control was not placed on a grid line originally, you will not be able to move it to a grid line with this option selected. For example, if you set the grid to be 4 × 4 and have a control with a position of (1, 1), when you move the control, it will only go to positions that are 4 units away in either dimension. Possible coordinates would be (5,5), (5,9), (9,5), and so on.

4. Check the Show Grid option and choose OK.

Changing the appearance of controls

You can change the appearance of a control using a Style dialog box. To bring up a Style dialog box for a control, double-click the control. The type of control determines which Style dialog box appears. For example, double-clicking a button control brings up the Button Style dialog box.

Although the Style dialog boxes differ from one another, they have many options in common. Once you understand how these options are used in one Style dialog box, you will know how to use them in others.

These options are common to most Style dialog boxes:

Table 4.5
Options common to Style dialog boxes

Option	Description
Caption	Lets you type the caption you want displayed with the control. Different type of controls display captions in different areas. For example, in a group box, the caption displays at the top left. In a push button, the caption displays inside the button. Not all controls display a caption. For example, a list box does not display the text specified in its caption. To the right of where you type the caption, check either Text or Number. Choose Text if you want the caption to be surrounded by quotation marks in the .RC or dialog file source code. Select Number if you don't want quotation marks. You can also add a caption to a control with the Caption window. See page 77.

Table 4.5: Options common to Style dialog boxes (continued)

Option	Description
Control ID	Lets you specify a unique identifier for the control. Control IDs can be a short integer or an integer expression. Type the control ID you want to assign to this control. By convention, static controls that are not modified at runtime are assigned a control ID of -1. If you type an alphanumeric identifier, Resource Workshop checks to see if a <i>#define</i> or a <i>constant declaration</i> has already been created for that identifier. If not, Resource Workshop asks if you want to create one. See Chapter 3 for more information about alphanumeric identifiers.
Scroll Bar	Lets you choose whether you want horizontal or vertical scroll bars included with your control.

Most controls have certain attributes in common:

Table 4.6
Control attributes

Attribute	Description
Tab Stop	Lets the user press <i>Tab</i> to access this control.
Group	Identifies the first control within a group. See page 80 for details about grouping and accessing controls.
Disabled	Dims the control to disable it (so the control does not respond to user input).
Border	Draws a border around the control (default).

Each type of control Style dialog box has options that are specific to a particular type of control. These are mentioned in the discussion of the various types of controls beginning on page 88.

Giving a control a caption

Although you can use a Style dialog box to add a caption to a control, you can also use the onscreen Caption window.

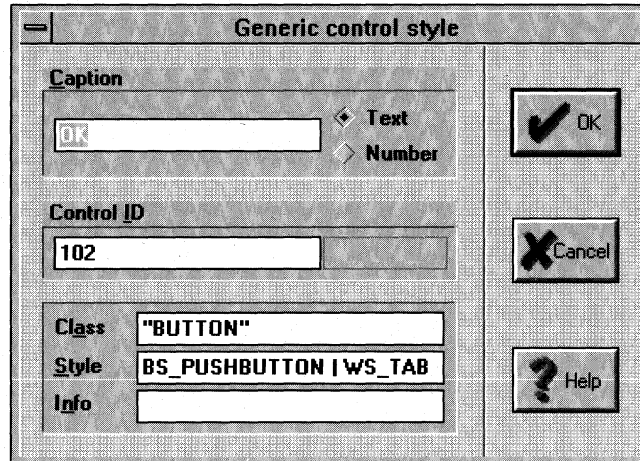
To add a caption to a control using the Caption window,

1. Select the control you want to add the caption to by tabbing to it or clicking it.
2. Type in a new Caption while the control is selected or press *F6* to move to the Caption window and type in a new caption.
3. Press *Enter*.

Changing a control's class

If you are working with custom controls, you may find the Generic Control Style dialog box helpful. Display it by holding down *Ctrl* and double-clicking your control. Or use *Tab* to select the control, then hold down *Ctrl* and press *Enter*.

Figure 4.10
The Generic Control Style dialog box



For more about custom controls, see page 99.

In the Generic Control Style dialog box, you can change the class of a control. You can also specify a caption, control ID, and style. If you type anything next to Info, the dialog box won't be compatible with the Microsoft Resource Compiler.

Specifying which controls are tab stops

By default, when you add controls in a dialog box, they are defined as tab stops. You can change this, so that only some controls are specified as tab stops. That way, users can use *Tab* to move only to controls you want.

There are three ways to change tab stops:

- Use the Tab Set tool
- Use the Style dialog box
- Use the Set Tabs command

Tab Set tool



To change a tab stop with the Tab Set tool,

1. Click the Tab Set tool. The cursor changes to the Tab Set icon. Resource Workshop surrounds any controls currently set as tab stops with boxes.
2. The Tab Set tool is a toggle:
 - To set a tab stop, click any control that is not surrounded by a box.
 - To remove a tab stop, click a control that is already a tab stop (surrounded by a box).
3. When you are through changing tab stops, click the Pick Rectangle tool so you can return to customizing your dialog box.

Style dialog box

You can also use a Style dialog box to change a tab stop:

1. Open the Style dialog box for the control (double-click the control or select it and press *Enter*).
2. Check Tab Stop under Attributes to set a tab stop, or uncheck Tab Stop to toggle the setting off.

Set Tabs command

A third way to set tab stops is to use the Set Tabs menu command:

1. Choose Options | Set Tabs. The mouse cursor turns into a Tab icon. Resource Workshop surrounds any controls currently set as tab stops with a box.
2. To change the Tab Stop attribute for any control, click it. The Set Tabs option works as a toggle, turning the Tab Stop attribute on or off.
3. When you finish setting tab stops, choose Options | Modify Controls. This returns you to edit mode.

You can test your dialog box, as described on page 101, to see how your new tab stops work.

Grouping related controls

You can define groups of controls. Then users can move within such groups using the arrow keys. It's common, for example, to group related radio buttons together, with only the first button in each group defined as a tab stop. Users can then press *Tab* to move to a new group, and use the arrow keys to move around within a group.

To define a group of controls,



1. If necessary, move the related controls so they're together.
2. Click the Set Groups tool. The mouse cursor becomes a G icon and any controls that are the first member of a group are surrounded by a box.
3. For each group you want to define, click the first member so it's surrounded by a box.

If any control that you don't want to define as the first member of a group is surrounded by a box, click it to remove its first-in-group setting.

You don't have to identify the last member of a group. By clicking the first member of each group, you also identify the last member of the previous group. If you want only one group, select the first member of your group. All the remaining controls will then become part of that group.

4. When you finish identifying the first member in each group, choose the Pick Rectangle tool. This returns you to edit mode.

You can also use the menu command *Options | Set Groups* to define groups. When you are through defining your groups, return to edit mode by choosing *Options | Modify Controls*.

You can test your dialog box, as described on page 101, to see if the groups work the way you think they should.

Reordering the controls

You can specify the order in which users can access the controls in a dialog box. The order is especially important when you've defined groups of related controls. If the controls aren't ordered properly, you'll end up with a confused user.

To specify the order of the controls in your dialog box,

1. Select the controls whose order you want to change. You can click and drag to select a group of controls.



You don't need to select any controls if you want to specify the order for all controls in the dialog box.

2. Click the Set Order tool. The mouse cursor turns into a Set Order icon.

Each control is numbered to show its current place in the overall order. If you chose just some of the controls in step 1, you'll see the order numbers only for those controls.

Note the **Next Item** prompt at the bottom of the Dialog editor. It tells you the order number that Resource Workshop assigns to the next control you click.

3. Click the items you want to assign new order numbers to. The Dialog editor displays a box around all the controls you've already picked.

While assigning new order numbers, you can "step back" by re-clicking the last control you just clicked. The order will change to its previous number. You can continue to backtrack by clicking the controls in the reverse order that you originally clicked them.

4. When you finish assigning new order numbers, click the Pick tool so you can continue editing. (You won't need to choose this command if you click all the selected controls.)

You can also order your controls with the menu command **Options | Set Order**. When you're done, choose **Options | Modify Controls** so you can continue to edit your dialog box.

See page 101 for details about testing a dialog box.

Once your controls are ordered the way you want them, test your dialog box to see if it works correctly.

Editing groups of controls

Once you've added controls to your dialog box, you can use the Dialog editor menu to align and resize groups of controls and put the controls in rows and columns.

Before you're ready to align and resize groups of controls, you need to know how to select more than one control at a time.

To select multiple controls,



1. Click the Pick tool to be in selection mode. This mode is the default, so you may be in selection mode already. (When selection mode is on, the Pick tool is selected and your cursor is the Pick icon.)

2. Click in the dialog box and continue to press your left mouse button. The position you click is the starting point for the sizing frame that appears as you drag your mouse.
3. Drag your mouse so that the sizing frame surrounds all the controls you want to select.
4. Release the mouse button. A box will appear around each selected control.

If you have selected a group of controls and want to add one or more controls to the group, or want to delete one or more controls to the group, hold down *Shift* and click the controls you want to add to or delete from the group.

Aligning multiple controls

The Dialog editor makes it easy to line up groups of controls with **Align | Align Controls**:

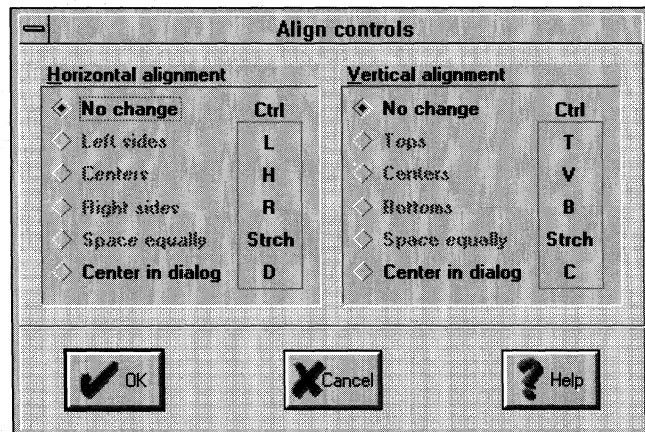
To align your controls,

1. Select the controls you want to align.

When you select a group of controls, note the sizing frame. It's wide enough and high enough to surround the outer edges of all selected controls. The alignment options move the controls within this border.

2. Choose **Align | Align Controls**. You'll see this dialog box:

Figure 4.11
The Align Controls dialog box



3. Choose from the Vertical Alignment and Horizontal Alignment options to move the selected controls.

Here's what the Horizontal Alignment options do. Most of these options are enabled only if you select two or more controls first.

Table 4.7
Horizontal alignment options

Option	Description
No Change	The controls won't move horizontally.
Left Sides	Moves controls horizontally so they are aligned along the left side of the sizing frame.
Centers	Moves controls horizontally so they are aligned in the center of the sizing frame.
Right Sides	Moves controls horizontally so they are aligned along the right side of the sizing frame.
Space Equally	Moves controls horizontally so they are spaced evenly within the sizing frame.
Center in Dialog	Moves the sizing frame horizontally so it's centered in the dialog box.

Here's what the Vertical Alignment options do. Most of these options are enabled only if you select two or more controls first.

Table 4.8
Vertical alignment options

Option	Description
No Change	The controls won't move vertically.
Tops	Moves controls vertically so they are aligned along the top of the sizing frame.
Centers	Moves controls vertically so they are aligned in the center of the sizing frame.
Bottoms	Moves controls vertically so they are aligned at the bottom of the sizing frame.
Space Equally	Moves controls vertically so they are spaced evenly within the sizing frame.
Center in Dialog	Moves the sizing frame vertically so that it's centered in the dialog box.

After you select the vertical and horizontal alignment you want, choose OK.

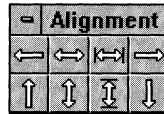
Note the accelerators, or short cut keys, listed in the Align Controls dialog box. Instead of using Align | Align Controls, you can select the group of controls and then use the Align Controls accelerators. For example, press *Ctrl+L* to align the group of controls horizontally along the left side of the sizing frame, or press *Ctrl+B* to align them vertically along the bottom of the sizing frame.



You can undo the alignment by choosing Edit | Undo, pressing *Alt+Backspace*, or selecting the Undo tool.

An even easier way to align controls is to use the Alignment palette. Select the controls you want to align by clicking and dragging. Then choose a tool from the Alignment palette.

Figure 4.12
The Alignment palette



Here are the tools you can choose:



Moves controls horizontally so they are aligned along the left side of the sizing frame.



Moves controls horizontally so they are aligned in the center of the sizing frame.



Moves controls horizontally so they are aligned along the right side of the sizing frame.



Moves the sizing frame and the selected controls horizontally so they are centered in the dialog box.



Moves controls vertically so they are aligned along the top of the sizing frame.



Moves controls vertically so they are aligned in the center of the sizing frame.



Moves controls vertically so they are aligned at the bottom of the sizing frame.



Moves the sizing frame and the selected controls vertically so they are centered in the dialog box.

The one alignment option that does not appear in the Alignment palette is the Space Equally option. Of course, you can always use the Align Controls dialog box, but there is another way you can space your controls equally using the sizing frame.

You can space controls equally by "stretching" the sizing frame.

To space your controls equally,

1. Select the group of controls you want to space equally. A sizing frame surrounds your selected controls.

2. Expand or “stretch” the sizing frame by holding down the *Ctrl* key while dragging the edge of the sizing frame in the direction you want to space your controls:
 - To horizontally space the controls equally, stretch a right or left border of the sizing frame.
 - To vertically space the controls equally, stretch a top or bottom border of the sizing frame.

See page 86 for details about using the Array dialog box.

If you stretch a corner of the sizing frame, the Array dialog box appears, ready for you to arrange your controls in rows and columns.

Resizing multiple controls

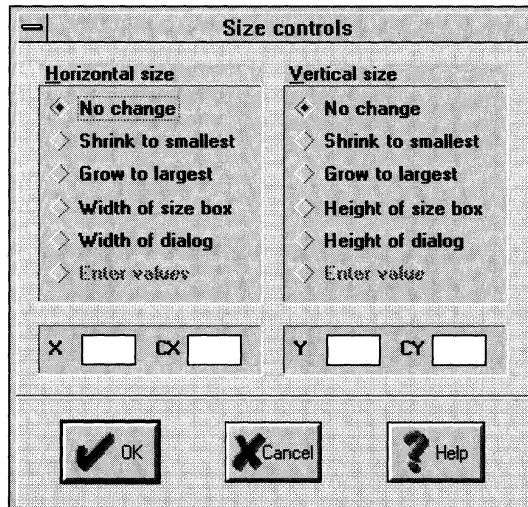
To resize a control, you can select it and drag the appropriate edge or corner. You can also resize groups of controls with the Size Controls option:

1. Select the controls you want to align.

When you select a group of controls, notice the sizing frame that surrounds all the controls. It’s wide enough and high enough to surround the outer edges of all the selected controls. The sizing options resize the controls within this border.

2. Choose **Align | Size Controls**. You’ll see this dialog box:

Figure 4.13
The Size Controls dialog box



3. Choose Vertical Size or Horizontal Size to move the selected controls.

Here's what the Horizontal Size options do. Most of these options are enabled only if you select two or more controls first.

Table 4.9
Horizontal size options

Option	Description
No Change	The controls won't change size horizontally.
Shrink to Smallest	Resizes controls horizontally so they are only as wide as the narrowest control in the selected group.
Grow to Largest	Resizes controls horizontally so they are as wide as the widest control in the selected group.
Width of Size Box	Resizes controls so they are as wide as the sizing frame.
Width of Dialog	Resizes controls so they are as wide as the dialog box.

Here's what the Vertical Size options do. Most of these options are enabled only if you select two or more controls first.

Table 4.10
Vertical size options

Option	Description
No Change	The controls won't change size vertically.
Shrink to Smallest	Resizes controls vertically so they are only as tall as the shortest control in the selected group.
Grow to Largest	Resizes controls vertically so they are as tall as the tallest control in the selected group.
Height of Size Box	Resizes controls so they are as tall as the sizing frame.
Height of Dialog	Resizes controls so they are as tall as the dialog box.

After you choose the vertical and horizontal sizing options you want, choose OK.

You can undo the sizing options by choosing Edit | Undo, by pressing *Alt+Backspace*, or by selecting the Undo tool.

Placing controls in columns and rows

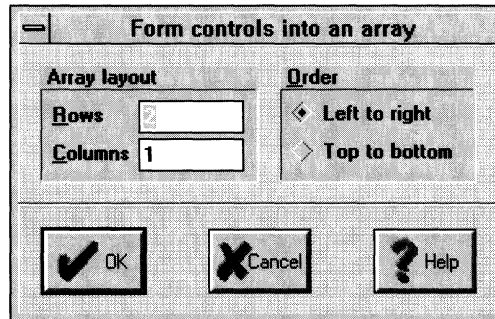
If you want to arrange controls in an array of columns and rows, use Align | Array. In addition to formatting the controls into columns and rows, Align | Array also numbers each control as a member of the array group.

Here's how to use the Array command:

1. Select the controls you want to arrange in columns and rows.
When you select a group of controls, note the sizing frame that surrounds all the controls. It's wide enough and high enough to surround the outer edges of all the selected controls.
2. If necessary, expand the sizing frame so the columns and rows you want will fit.
3. Choose **Align | Array** or press the Duplicate tool. You'll see this dialog box:



Figure 4.14
The Form Controls Into An
Array dialog box



4. Under Array Layout, specify the number of rows and columns you want.
5. Under Order, select how you want to order the controls in this group. For example, suppose you are arranging nine controls into three columns.

If you choose Left to Right, here's how the Dialog editor numbers the controls:

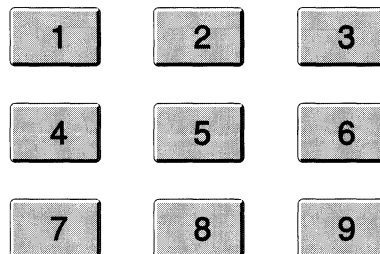
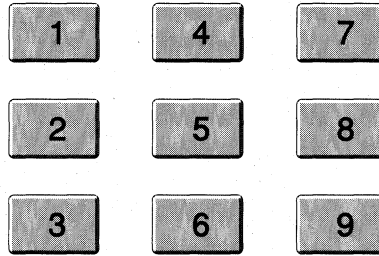


Figure 4.15
Controls in left to right order

Figure 4.16
Controls in top to bottom
order



If you choose Top to Bottom, here's how the Dialog editor numbers the controls:

6. After you've chosen the options for the array, choose OK.

Undoing changes



You can “undo” any editing you do in the Dialog editor, such as placing controls, aligning them, deleting controls, and so on, with the Undo tool or the Edit | Undo command. Undo works on commands that affect groups of controls as well as commands that change single controls. See Chapter 3 for details about using Undo.

Button controls

Radio buttons, push buttons, check boxes, and group boxes are all types of button controls. Although you can add them to your dialog box with the Tools palette, you need to use the Button Style dialog box to modify existing button controls. To display the Button Style dialog box, double-click the button control you want to modify.

Change the button type by choosing a new option under Button Type.

Table 4.11
Button types

Button type	Description
Push Button	A button containing text. The user clicks the button, which sends a BN_CLICKED message to the parent window.
Default Push Button	Identical to a push button, but also includes a bold border indicating that it's the default response if the user presses <i>Enter</i> .

Table 4.11: Button types (continued)

Button type	Description
Check Box	A rectangular button that can include text to the left or right of the button. The box is marked with an X when it's selected. It is the application's responsibility to check and uncheck the box when the user uses a check box.
Auto Check Box	Identical to a check box, but Windows does the checking and unchecking instead of the application.
3-state	Identical to a check box, but includes a third possible state: The button can be dimmed to show that its state is unknown or indeterminate. It is the application's responsibility to check, uncheck, and dim the box.
Auto 3-state	Identical to a 3-state button, but Windows does the checking and unchecking instead of the application program.
Radio Button	A circular button that has identifying text to the left or right. The circle is filled with a solid dot when it's been selected. It is the application's responsibility to fill or clear the dot when the user selects a radio button. Radio buttons must appear in groups. Usually, a group of radio buttons presents the user with a set of mutually exclusive options. When the user clicks a radio button, it sends a <code>BN_CLICKED</code> message to the parent window.
Auto Radio Button	Identical to a radio button, but Windows does the filling in or clearing of the dot instead of the application.
Group Box	A rectangular box used to group buttons together. You can also include a caption to display in the upper left corner of the group box.
User Button	Customized buttons for Windows 2.0 compatibility; we recommend that you don't use user button controls with Windows 3.0. Instead, you should use owner draw buttons.
Owner Draw	A button that allows the application to paint the button itself. When the button needs painting, it sends a <code>WM_DRAWITEM</code> message to its parent.

The Justification options determine if the text for the radio and check box buttons appears to the left or right of the button.

Table 4.12
Justification options

Option	Description
Left	Justifies text to the left of the button.
Right	Justifies text to the right of the button.

Scroll bar controls

A scroll bar is used to represent a range of values. A scroll bar is a rectangle with direction arrows at each end. Between the arrows, a square icon (sometimes called a *thumb*) indicates the current value.

You can add vertical and horizontal scroll bars anywhere you want in a dialog box.

Although you can place scroll bars using the Tools palette or Controls | Scrollbar, you must customize your scroll bars with the Scroll Bar Style dialog box. To display it, double-click the scroll bar control you want to modify.

The Scroll Bar Style dialog box includes the common and the control-attribute options listed in the tables beginning on page 76, as well as alignment options that align the scroll bar.

Table 4.13
Alignment options

Option	Description
None	The scroll bar fills the entire rectangle (default).
Top Left	The scroll bar displays in the upper left corner of the rectangle.
Bottom Right	The scroll bar displays in the lower right corner of the rectangle.

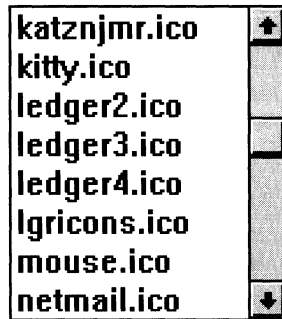
List box controls

A list box is a rectangle containing a list of text strings. Usually, a user can browse through what's displayed in a list box, then select one or more items. The list box sends a message to the parent window about the selected item(s).

You'll often see list boxes used in a File Open dialog box.

Figure 4.17
A list box in a File Open
dialog box

Files:



If the list of items exceeds the length or width of the list box, you can add scroll bars to the list box.

Other than the common options described on page 76, the List Box Style dialog box has Owner Drawing and List Box options.

Owner Drawing options let you determine whether the list contained in the list box should be drawn by the list box or the application. Choose one of the attributes in this table:

Table 4.14
Owner Drawing options

Option	Description
Not Owner Draw	The list box control draws the list (default).
Fixed	The application draws the list box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages. A list box control sends a WM_MEASUREITEM message to the application only when the list box is initially drawn, which fixes the list box item height.
Variable	The application draws the list box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages. The list box control sends the WM_MEASUREITEM message to the application for each item in the list box; each item can vary in height.
Has Strings	If you've chosen either Fixed or Variable, the list box stores text for each list item with the LB_INSERTSTRING or LB_ADDSTRING. The list box can also retrieve list items from the message LB_GETTEXT.

List Box options let you further define the list box. Choose one or more of the options in this table:

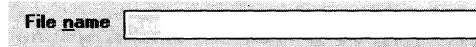
Table 4.15
List Box options

Option	Description
Notify	Sends an input message to the parent window when the user clicks on an item in the list (default).
Sort	Sorts the list alphabetically.
Multiple Select	Lets the user select more than one item at a time. The user can also toggle individual items on and off.
Don't Redraw	Prevents the list box from being redrawn when it is changed.
Tab Stops	Organizes the information in the list box in columns. The default column width is 32 dialog units or 8 characters. You should use <i>Tab</i> characters (\x09) to format the text. (If you want to change the column width, the application should set its own tab stops using the LB_SETTABSTOPS message.)
Integral Height	Causes the list box to resize its height at runtime so the client area is large enough to display items completely (default). If Integral Height is turned on and the list box needs to be resized to show items completely, the list box decreases its size. For example, if three items are completely displayed at runtime but another item doesn't quite fit, the list box will decrease its size so that only the three items are displayed.
Multi Column	Lets the user horizontally scroll the list box to work with multiple columns. If you turn on this option, the application must send the LB_SETCOLWIDTH message to set the width of all columns in pixels.
Pass Keyboard Input	Passes what the user types to the application.
Extend Select	When this option is used with multiple select list boxes, this attribute modifies the way multiple selection works, so that the user can select more than one item in the list.

Edit text controls

An edit text control lets the user enter text from the keyboard. A common use of an edit text control is found in a File Open dialog box.

Figure 4.18
An edit text control from a
File Open dialog box



To customize an edit text control, double-click it in the dialog box. The Edit Text Style dialog box appears.

The Edit Text Style dialog box includes the common and control-attribute options listed in the tables beginning on page 76, as well as the following options:

- Justification
- Automatic Scroll
- Line
- Case
- Other

The Justification options determine how text typed into an edit text control appears.

Table 4.16
Justification options

Option	Description
Left	Justifies multiple-line text to the left (default).
Right	Justifies multiple-line text to the right.
Center	Centers multiple-line text.

The Automatic Scroll options control automatic scrolling.

Table 4.17
Automatic Scroll options

Option	Description
Horizontal	When the user types a character at the right edge of the edit text boundary, the text automatically scrolls ten characters to the right. When the user presses <i>Enter</i> , the text scrolls back to the zero position.
Vertical	When the user presses <i>Enter</i> on the last line of the edit text control, the text scrolls up a full page. For example, if the control is five lines long, pressing <i>Enter</i> on the last line causes text to scroll up five lines; the cursor goes back to the top line. For this option to have any effect, you must also define the edit text control to allow for multiple lines.

Line options determine if one or more lines can be entered in the edit text control.

Table 4.18
Line options

Option	Description
Single Line	Limits the edit text to a single line (default).
Multiple Line	Lets the user type text on multiple lines. (You'll probably want to choose how the text will be scrolled; see the Vertical Automatic Scroll option.)

Case options determine if the text the user types in the edit text control displays in uppercase, lowercase, or as it is typed.

Table 4.19
Case options

Option	Description
Case Insensitive	Displays text exactly as typed (default).
Upper Case	Displays all text in uppercase letters, regardless of how it's typed.
Lower Case	Displays all text in lowercase letters, regardless of how it's typed.

Here are three more options available to edit text controls:

Table 4.20
Other options

Option	Description
Password	When Password is on, the display of each letter as it is typed is suppressed. Instead, an asterisk appears in its place. This is helpful for keeping passwords secret.
Convert OEM	Converts text typed into the control to the current OEM character set, then reconverts the text to ANSI. This option is useful in file input boxes because it ensures that any file name entered will be translatable into the OEM character set, which is used by the DOS file system.
Keep Selection	Highlights selected text, even when this control doesn't have the keyboard focus. For example, suppose the user highlights half the text displayed in an edit text control. Normally, when the user moves to another control the text is no longer highlighted. Choose Keep Selection when you want the user to see the highlighted text even when working with other controls.

Static controls

A static control displays text or art the user can't change. You can use static controls to label portions of your dialog box or to present information graphically.

The static controls in the Tools palette include

- Static text
- Icon
- Frame
- Rectangle

You can customize static controls with the Static Style dialog box. Double-click the static control in the dialog box and the Static Style dialog box appears.

The Static Style dialog box includes the common and control-attribute options listed in the tables beginning on page 76, as well as several options specific to static controls.

For an Icon static control, the Caption field must contain the name or the numeric ID of the icon resource or the icon will not display. If the Icon resource has a name, enter the name and select the Text option. If the icon resource has no name, enter the number and select the Number option. See page 97 for more details.

The No Character Underline check box lets you turn off character underlining. If your static control includes text, you can underline a character by preceding it with an ampersand (&). Check No Character Underline if you want to disable underlining. Uncheck it when you want underlining enabled.

Control Type options let you further define what is displayed by the static control. Choose from the following options:

Table 4.21
Control Type options

Option	Description
Left Text	Displays text flush left within the control border (default). If there are multiple lines of text, words that extend beyond the line wrap to the next line.
Left Text—No Wrap	Displays text flush left within the control border. If there are multiple lines of text, this option automatically disables wordwrapping. You must end lines with a carriage-return character.
Centered Text	Displays text centered within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.
Right Text	Displays text flush right within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.
Simple Text	Displays a single line of flush-left text.
White Rectangle	Displays a filled-in rectangle that's invisible because it's the same color as the current window background. The default Windows color for the window background is white.
Gray Rectangle	Displays a filled-in rectangle that's the color of the current screen background (desktop). The default Windows color for the desktop is white.
Black Rectangle	Displays a filled-in rectangle that's the same color as the current window frame color. The default Windows color for window frames is black.
White Frame	Displays an empty frame that's invisible because it's the same color as the current window background. The default Windows color for window backgrounds is white.
Gray Frame	Displays an empty frame that's the color of the current screen background (desktop). The default Windows color for the desktop is gray.

Iconic static controls

Resource Workshop lets you display icons in a dialog box. To place an iconic static control in your dialog box,

1. Double-click on the iconic static control to display the Static Style dialog box.
2. In the Caption field, type in either the name or the numeric ID of the icon resource. If you enter a name, select the Text option. If you enter a number, select the Number option. For example, if you have an icon named GLOBE in your project file that you want to display in your dialog box, type `GLOBE` in the Caption input box and select the Text radio button. Choose OK. Your icon appears in your dialog box.



Remember that if you enter an identifier in the Caption field, you should select the Number option because an identifier evaluates to a number.

3. If you want to edit the icon, double-click the icon to display the Static Style dialog box once again. The Edit Icon button is now enabled. Click it to start the Paint editor so you can edit the icon.

Combo box controls

A combo box is just what the name implies: a combination of two different types of controls. A combo box combines a list box (a control that lets the user browse and select strings) with either a static control (text a user can't change) or an edit text control (an area where a user can type).

Figure 4.19
A combo box from a File
Open dialog box



You can customize a combo box using the Combo Box Style dialog box. Double-click the combo box in the dialog box and the Combo Box Style dialog box appears.

Type options let you define the combo box. Choose from these options:

Table 4.22
Combo box type options

Option	Description
Simple	The drop-down list is always expanded to display items in the list, and the user can edit the items in the list (default).
Drop Down	When the dialog box is first displayed, the combo box consists of a single line of editable text. The user can click the down arrow to expand the list, and edit all items in the list.
Drop Down List	Works just like a drop down, but the list is static. The user can select, but can't change anything in the list.

Owner Drawing options let you determine whether the list contained in the list box should be drawn by the list box itself or by the application. Choose one of these options:

Table 4.23
Owner drawing options

Option	Description
No	The list box control draws the list (default).
Fixed	The application draws the list box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages. The list box control sends the WM_MEASUREITEM message to the application only when the list box is initially drawn, which fixes the list box size.
Variable	The application draws the list box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages. The list box control sends the WM_MEASUREITEM message to the application for each item in the list box; the list box can therefore vary in size.
Has Strings	If you've chosen either Fixed or Variable, the list box stores text for each list item with the LB_SETTEXT message. The list box can also retrieve list items from LB_GETTEXT.

The Combo Box Style dialog box includes the common and control-attribute options listed in the tables beginning on page 76, as well as options specific to combo box controls.

Table 4.24
Combo box attributes

Option	Description
Vertical Scroll	Puts a vertical scroll bar in the list box.
Sorted	Automatically sorts items in a list box in alphabetical order.
Integral Height	Sizes the list box at runtime so all items in the list are completely displayed (default). If you need to precisely control the height of the list box, uncheck this option.
OEM Conversion	Converts text the user types in to the current OEM character set, then reconverts the text to ANSI. This option is useful in file input boxes because it ensures that any file name entered will be translatable into the OEM character set, which is used by the DOS file system.
AutoHorizontal	Scrolls text to the left automatically when it exceeds the width of the control.

Custom controls

If you want to use a control that doesn't fit into one of the predefined Windows 3.0 types, you can use a *custom control*. The predefined controls discussed earlier in this chapter, such as list boxes, scroll bars, and buttons, are called standard controls. They were developed by Microsoft and are part of Windows. A custom control, on the other hand, is any other window class that you want to include in your dialog boxes.

There are two types of custom controls: those that you can install and those that are application-specific. You must implement installable custom controls using a dynamic-link library. Custom controls specific to an application are implemented in the application itself. Resource Workshop draws them as either gray boxes or empty frames.

Creating your own custom controls

If you want to create your own custom controls, you'll need to design them and store them in dynamic-link library (DLL) files. Look in the README file on your distribution disks for details about creating custom controls.

Installing a custom control library

Custom controls are stored in DLLs. If you want to add custom controls to your dialog box, you'll need to install the appropriate DLL file(s). Then the custom controls in that DLL will be available just like any standard Windows control.

To install a DLL file containing a custom control library,

1. In the Dialog editor, choose Options | Install Control Library. You'll see the Install a New Control Library dialog box.
2. Specify the custom control DLL file.
3. Choose *OK*.

Now the controls contained in that DLL file are available for you to add to your dialog box.

Displaying custom controls

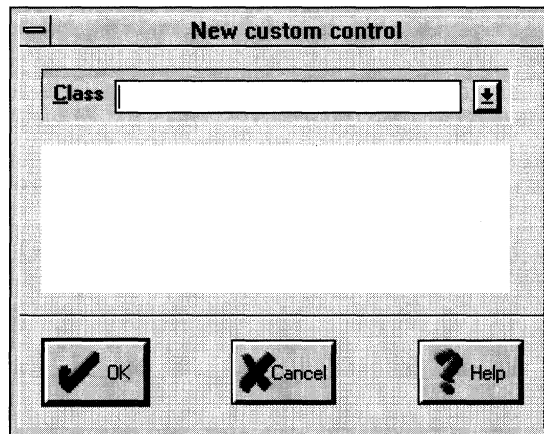
Before you add any custom controls to your dialog box, decide how they'll be displayed in the Dialog editor. Choose Options | Preferences to see if your dialog boxes will display in Draft or Normal mode. If one of these two modes is selected, you might want to uncheck the "Draw Custom Controls as Frames" option so you'll see more than just a frame to represent your custom controls.

Adding a custom control

Once you've installed a DLL file containing custom controls, you can add any of those custom controls to your dialog boxes.

1. Click the custom control tool or choose Control | Custom. You'll see the New Custom Control dialog box.

Figure 4.20
The New Control Custom
dialog box



2. From the drop-down list next to Class, choose the custom control you want. Resource Workshop displays a sample of the custom control you've selected in the middle of the dialog box.
3. When you've selected the custom control you want, choose OK. The mouse cursor becomes a cross hair, indicating that it's ready to place the custom control.
4. Click in the dialog box window where you want to place the custom control.

Testing a dialog box



The Dialog editor makes it easy to test your dialog box so you can see the effect of any changes you've made. Select the Test tool or choose Options | Test Dialog to test your dialog box.

Now you can press *Tab* and the arrow keys to see how you can move around your dialog box, or you can type text to see how text is scrolled in an edit text control. Check to see if your controls are in the order you want them.

When you test a dialog box, the status line at the bottom of the Dialog editor says *Test*.

When you want to leave test mode, choose Options | Test Dialog again, simply press *Enter*, or select the Pick Rectangle tool to return to edit mode.

Saving a dialog box

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. There are two primary options you can use to save changes you've made to a dialog box. You can

- Save the entire project
- Save the dialog box in a dialog resource script file (.DLG)

Saving the project

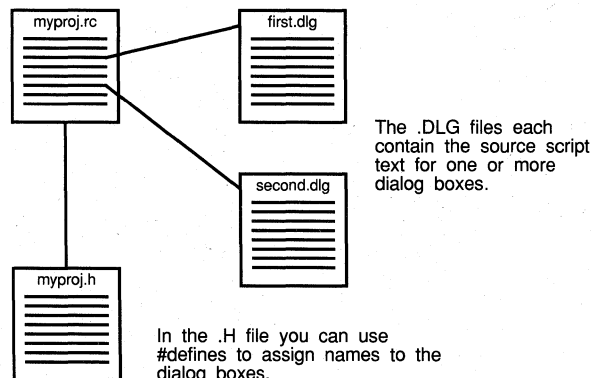
To save the entire project, choose File | Save Project. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. It also updates the dialog box in an external dialog resource script file.

Saving a dialog box in a file

You can also store your dialog box resources in dialog resource script files (usually files with a .DLG extension). A dialog file usually contains script defining one or more dialog box resources, but can contain other resources also. If you add a dialog file to a project, Resource Workshop automatically adds an **rcinclude** reference in the Project window, which references a dialog file.

Figure 4.21
The .RC file points to .DLG
and .H files

The .RC file can contain the source script text for one or more dialog boxes. It can also contain **RCINCLUDES** to reference .DLG files.



To store dialog boxes in a dialog file (.DLG), you'll need to add a dialog file to the project. Then you can choose to store dialog

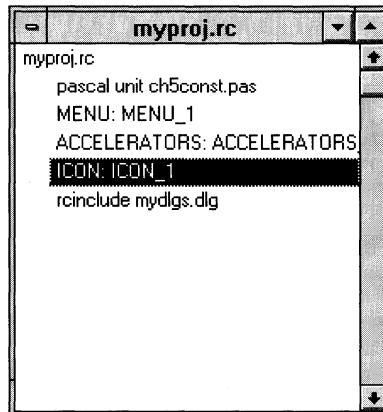
boxes in this dialog file as you create them. To add a dialog file to a project and store a new resource in it,

See Chapter 3 if you need more information about projects.

1. Choose File | Add to Project. The Add to Project dialog box appears.
2. In the File Type drop-down list box, select DLG Resource Script.
3. Type a new file name in the File Name input box.
4. Choose OK. A dialog box appears with the message, "<filename> does not exist. Create it?"
5. Choose Yes.

An **rcinclude** statement is added to your Project window. If you can't see it, choose View | By File.

Figure 4.22
A project window, with
rcinclude for a .DLG file



You've created an empty dialog file. Now you need to create a new dialog box and add it to the dialog script file.

1. Choose Resource | New. A New Resource dialog box appears.
2. Select the dialog script file you just created in the "Place resource in" combo box.
3. In the Resource Type list box, double-click DIALOG, or select DIALOG and choose OK.

You'll see your new dialog box in the Dialog editor ready for you to customize with controls. When you are through customizing your dialog box and exit the Dialog editor, you'll see a DIALOG entry indented under the **rcinclude** for the .DLG file in the Project window.

Viewing two dialog boxes

Resource Workshop gives you a way to compare or otherwise view two dialog boxes at the same time.

To view two dialog boxes,

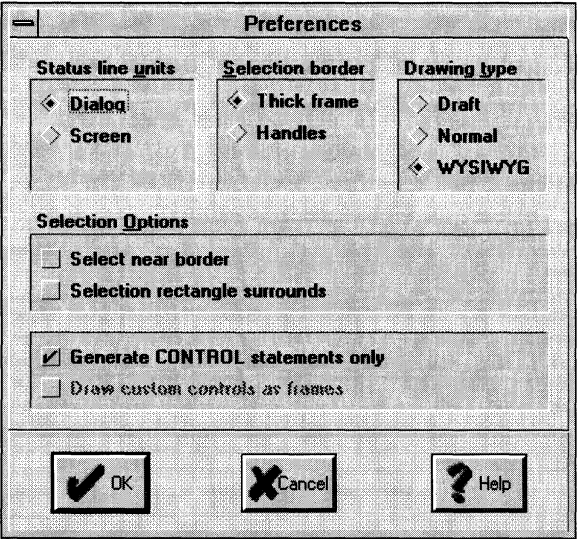
1. In the Project window, click one of the dialog boxes you want to view. The Dialog editor starts up and displays that dialog box.
2. Click the Test tool or choose Options | Test Dialog.
3. Click the Minimize button of the Dialog editor. The Dialog editor won't shrink to an icon as you would expect. Instead, your dialog box is now a floating, modeless dialog box you can move around as you like.
4. Return to the Project window and click the second dialog box you want to view. A second Dialog editor starts up.
5. Again, click the Test tool or choose Options | Test Dialog in the second Dialog editor.
6. Click the Minimize button of the second Dialog editor.

Now you have two floating dialog boxes you can put side by side.

Customizing the Dialog editor

Resource Workshop lets you change the way some parts of the Dialog editor work. Choose Options | Preferences to display the Preferences dialog box.

Figure 4.23
The Preferences dialog box



Status line units determine the unit of measurement the status line uses to display information.

Table 4.25
Status line units

Unit	Description
Dialog	Uses the dialog unit as the unit of measurement on the status line. In a dialog unit, y equals $\frac{1}{8}$ of the font height, and x equals $\frac{1}{4}$ of the font width.
Screen	Uses a pixel as the unit of measurement on the status line.

The Selection Border options let you change the appearance of the box that surrounds controls when they are selected.

Table 4.26
Selection Border options

Option	Description
Thick frame	The selection boxes appear as thick frames around the selected controls (default).
Handles	The selection boxes that appear as frames around the selected controls have handles.

For more information about the resource script language, use the Help system.

In the resource script language, each type of dialog control has a unique syntax. For example, centered static text uses the CTEXT statement. The CONTROL statement, however, can specify any type of dialog control. If you want your Resource Workshop to

generate only CONTROL statements in your resource script rather than the specialized dialog control statements, select the Generate CONTROL Statements Only option.

Drawing Type options determine how elements of your dialog box are displayed in the Dialog editor.

Table 4.27
Drawing Type options

Drawing Type	Description
Draft	Draws controls as rectangles with the order number centered within. Using this drawing type is the easiest way to see the exact sizes of control rectangles.
Normal	Simulates the drawing of controls. This is the fastest drawing type option, but custom controls are drawn as gray boxes only.
WYSIWYG	With this option selected, Resource Workshop creates the dialog and control child windows and the controls draw themselves. This option is slower, but the most accurate. Installable custom controls draw themselves. This is the default option.

Selection options determine how the selection feature behaves when you select controls to customize.

Table 4.28
Selection options

Option	Description
Select Near Border	When this option is on, you must click very close to the border of a control to select it. If this option is not selected, just clicking in the vicinity of the control will select it. If you are working with closely spaced controls, you may want to turn this option on to get finer control and to avoid selecting controls inadvertently.
Selection Rectangle	When this option is on, you must entirely surround the controls you want to select with the selection rectangle or the controls will not be selected. If this option is not checked and the selection rectangle surrounds only part of a control, that control will be selected. If you are working with closely spaced controls, you may want to turn this option on to get finer control and to avoid selecting controls inadvertently.

A sample project

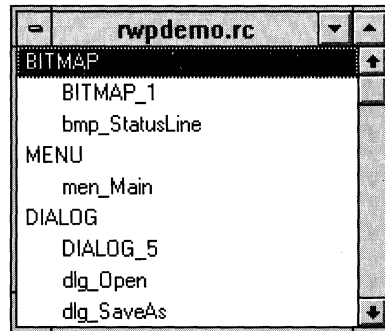
Resource Workshop comes with a sample project, RWPDEMO.RC, that will give you a taste of working with the Dialog editor. Follow these steps:

1. Choose File | Open Project.
2. Use the Open File dialog box to find and open the RWPDEMO.RC file that came with Resource Workshop.

In the File Type box, choose RC Resource Script, then use the Files and Directories list boxes to find RWPDEMO.RC. Or you can type the complete path name next to File Name.

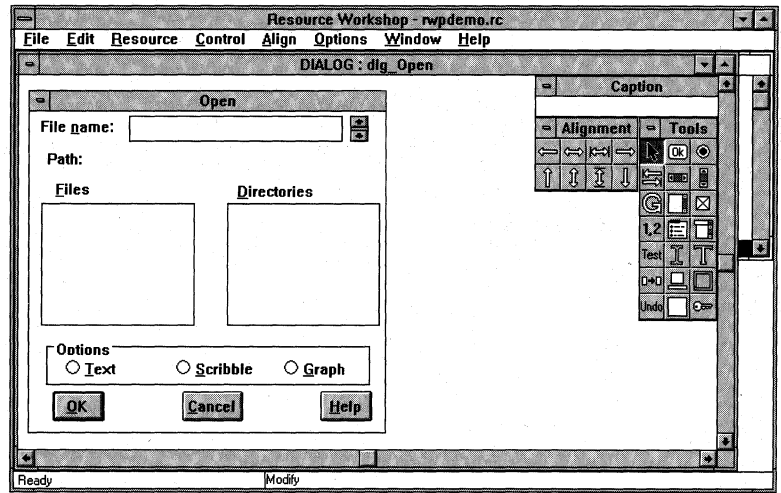
After you open RWPDEMO.RC, you'll see the Project window.

Figure 4.24
The Project window for
RWPDEMO



3. In the Project window, double-click the `dlg_Open` entry.
Now you'll see the Dialog editor with an Open dialog box defined in it.

Figure 4.25
The sample Open dialog box



Notice the Tools palette on the right side of the Dialog editor. It contains different types of controls that you can add into the dialog boxes you create.

4. Double-click in the center of the list box under Files.

You'll see a List Box Style dialog box where you can specify more detailed information about the list box control. Note that a vertical scroll bar has been selected for this list box.

In general, you can double-click any control in your dialog box if you want to specify detailed style options. If you prefer using the keyboard, you can use *Tab* to select a control, then press *Enter* or choose *Control | Style* to display the Style dialog box.

5. Choose *Cancel* to close the List Box Style dialog box.
6. If you want, you can experiment a little. Double-click other controls to see what the different Style dialog boxes look like.
7. To see the Style dialog box for the dialog box window, double-click the outer edge or the title bar of the window. Or you can use *Tab* to select a window, then press *Enter* or choose *Control | Style* to display the Style dialog box.
8. You can also test the dialog box to see how it will work at runtime. If a Style dialog box is open, close it first. Click the *Test* tool or choose *Options | Test Dialog*.

You'll see a test version of the dialog box. You can experiment by doing such things as typing text next to File Name, pressing

Tab to see how you can move through the controls, and opening the Control menu in the upper left corner. Press *Enter* or *Esc* to leave test mode. You can also leave test mode the same way the user will exit the dialog box: by choosing OK or Cancel, or by closing the dialog box using the Control menu button and choosing Close.

To exit the RWPDEMO.RC project, choose File | Close All.

Creating a new dialog box

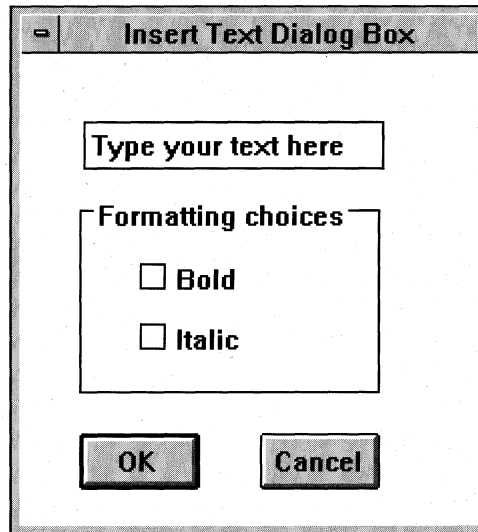
You've seen how to work with an existing dialog box; now you'll create a new dialog box. But before you do, you might want to consider these questions:

- What type of window should the dialog box be in? Should it include a caption on the title bar? Should it be a pop-up window? Do you need a Control menu?
- What types of controls do you need?

For mutually exclusive choices, you may want to use a group of radio buttons. If users need to choose a file, you might want to include both a list box and an edit text control, so users can choose to type the file name or select it from a list.
- Once you've added all the controls you need, how should they be arranged in the dialog box? Will users have access to all controls using *Tab*?

Suppose you want to create a dialog box where users can type text and make a couple of formatting choices. Here's the design for the dialog box:

Figure 4.26
The sample dialog box



Starting from the top, let's look at the different parts of this dialog box.

- The dialog box includes a caption on its title bar that identifies it: Insert Text Dialog Box.
- The box that says "Type your text here" is an edit text control. Users can type in their own text.
- The "Formatting choices" box is really three different controls: a group box control surrounding two check box controls.
A *group box* is a frame that groups buttons together. The check boxes, Bold and Italic, are choices that the user turns on and off by selecting them.
- The OK and Cancel buttons are push button controls. The OK button is the default choice (it's chosen if the user presses *Enter*).

If you had to write the resource script code yourself to create this sample dialog box, this is how the code would look (the numbers in the source code would vary depending on the size and placement of the dialog box window and controls):

```

MY_DB DIALOG 18, 18, 115, 118
CAPTION "Insert Text Dialog Box"
STYLE DS_MODALFRAME|WS_POPUP|WS_CAPTION|WS_SYSMENU
BEGIN
    CONTROL "Type your text here", 102, "EDIT",
        ES_LEFT|WS_BORDER|WS_TABSTOP, 16, 17, 75, 12
    CONTROL "Bold", 2, "BUTTON",
        BS_CHECKBOX|WS_TABSTOP, 30, 50, 28, 12
    CONTROL "Italic", 3, "BUTTON",
        BS_CHECKBOX|WS_TABSTOP, 30, 65, 28, 12
    CONTROL "Formatting choices", 4,
        "BUTTON", BS_GROUPBOX, 15, 35, 75, 50
    CONTROL "OK", 106, "BUTTON", BS_DEFPUSHBUTTON|WS_TABSTOP,
        15, 95, 30, 14
    CONTROL "OK", 107, "BUTTON", BS_PUSHBUTTON|WS_TABSTOP,
        60, 95, 30, 14
END

```

Now let's see how easy it is to create the dialog box using Resource Workshop's Dialog editor.

Starting the Dialog editor

First, open a new project and start the Dialog editor. (See Chapter 3 if you need information about opening a project.)

1. Choose File | New Project.
2. In the New Project dialog box, select .RC and choose OK.
3. Choose Resource | New. The New Resource dialog box appears.
4. In the Resource Type list box, double-click DIALOG, or select DIALOG and choose OK.

Customizing the dialog box

You're now in the Dialog editor. Here's how to customize your new dialog box:

1. Select the empty dialog box window by clicking its title bar. You'll see a selection border around the window.
2. Type `Insert Text Dialog Box` for the dialog box caption. Press *Enter*.

Adding the text control

Next add the edit text control.



1. From the Tools palette, choose the Edit Text tool by clicking it once. Move to the dialog window and add an edit text control

in it by clicking your left mouse button when the Edit Text icon is where you want the control to appear.

When you add a new control, the sizing frame is displayed around it, which means it's selected. Drag a corner or edge to make the edit control larger, and drag from within the control to move it.

2. While your new edit control is selected, type *Type your text here* and press *Enter*.

Adding the check boxes

Now add the two check boxes and surround them with a group box. The Check Box tool is the third tool in the right column of the Tools palette.



1. With the left mouse button, select the Check Box tool from the Tools palette by clicking it, then move the mouse pointer to where you want to place the check box in the dialog box window and click your mouse again. You can place additional copies of the previously selected control with the right mouse button, so position the pointer to where you want the second check box to appear, and click the right mouse button.
2. Add the Bold and Italic captions that will appear next to the check boxes by selecting each control and typing the caption text.



3. The Group Box tool is in the center of the left column of the Tools palette, right above the Edit Text tool. Add the group box control in the dialog box. Move and resize it as necessary so that the group box control encompasses the check boxes.
4. While the group box is selected, type *Formatting choices* and press *Enter*.
5. If you need to, move things around and resize them until the dialog box is arranged so that everything fits and looks OK.

Adding the push buttons

Now you're ready to add two important push buttons: OK and Cancel.

1. If you don't have room in your dialog box for these controls, make your dialog box larger. Click the title bar of the dialog box. Drag the bottom border of the dialog box until it is the size you want.
2. From the Tools palette, select the Push Button tool.

3. Click in your dialog box where you want to place your push button.
4. Move to where you want the second push button to appear and click the right mouse button. A copy of the push button appears.
5. Select the first push button and type `OK`.
6. Double-click the `OK` button. The Button Style dialog box appears.
7. Select the Default Push Button button type and choose `OK`.
8. Select the second push button, type `Cancel`, and press `Enter`.
9. If you aren't happy with the way your dialog box looks, move and resize its controls until you are satisfied.

Testing the dialog box Now test the dialog box to see how it works.



1. Click the Test tool.
Type new text in the edit text control, and experiment with the *Tab* key to see how you move around the dialog box.
2. When you are done, press `Enter` to leave test mode. Now you can make any changes you need.

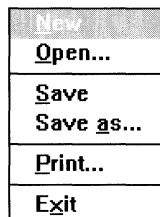
Saving the project When you are finished with your dialog box, save the project as `MYPROJ.RC`:

1. Choose `File | Save File As`.
2. In the New File Name input box, type `MYPROJ`.
3. Choose `OK`.

Creating menus

Most Windows applications include a menu bar. Each choice in the menu bar produces a *pop-up* menu (also known as a *drop-down* menu) that shows another list of choices. For example, most Windows programs include a File menu that includes options for creating a new file or opening an existing file.

Figure 5.1
A typical File menu



See the documentation that comes with your compiler or the Microsoft Windows Programmer's Reference for more information on using the `TrackPopupMenu` function.

Most pop-up menus are attached to the initiating menu command. However, you can make a pop-up menu appear anywhere on the application window by using the `TrackPopupMenu` function. This type of pop-up menu is called a *floating* menu.

Resource Workshop provides you with a Menu editor that makes it easy to create and edit menus for your application. When working with menus, you perform the following four primary tasks:

- Starting the Menu editor
- Customizing a menu
- Testing the menu

■ Saving the menu

See page 135 for an example illustrating how to use the Menu editor.

The first task, starting the Menu editor, displays a menu ready for you to customize. The second and third tasks, customizing the menu and testing it, you perform using the Menu editor. The fourth task, saving the menu, happens automatically when you save your project.

Before discussing these tasks, we show you how the Menu editor screen and features are organized. The next sections cover the four primary tasks you perform in creating a menu, followed by a description of how to save changes, a description of how to edit a Menu resource script, and an example of how to create a menu.

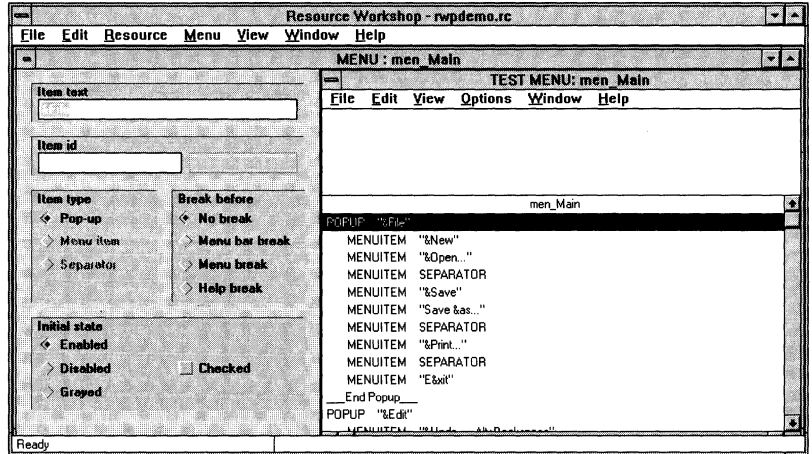
If you like, you can open the demo project RWPDEMO.RC, click on the Menu resource, and try features out as you read about them in this chapter.

The Menu editor screen

The Menu editor uses three panes to display editing information: an Outline pane that's similar to source script, a Test Menu pane, and a Dialog Box pane (usually called the dialog box) that lets you customize the currently highlighted line in the outline. You can change pane positions using the View command, and you can create new menu entries using the Menu command.

Resource Workshop displays the following menu if you open the demo project RWPDEMO.RC:

Figure 5.2
The Menu editor with
RWPDEMO's menu



The Outline pane

The Outline pane, on the bottom right, shows you the outline of the new menu. When you add menu items, pop-up commands, and separators, they appear in pseudocode in this pane. The top line in the pane is the name of the menu, and the other lines are *statements* defining a single pop-up menu containing a single menu item. To edit a menu entry, select a statement in this pane.

To select any line in this pane, you can use the mouse or you can do the following:

- If you're in this pane, you can use $\uparrow$, $\downarrow$.
- From anywhere in the window, you can use $Ctrl+\uparrow$, $Ctrl+\downarrow$.
- With the mouse, you can select a menu command in the test menu, which moves you to that same command in the outline.

You can use $F6$ or $Shift+F6$ to move between the dialog box and the Outline pane.

The outline you see on the Menu editor doesn't show all parameters for each statement. Instead, it shows you pseudocode that's designed to make it easy to work with the structure of your menu. If you want to see the complete code, edit the menu's resource script (see "Editing a menu resource script" on page 133).

The Dialog Box pane

The Dialog Box pane (dialog box for short), on the left side of Figure 5.2, lets you specify information about the statement that's currently highlighted in the outline. Your choices define the characteristics of the highlighted menu statement.

To move around inside the dialog box, you can use the mouse to position anywhere and make selections. You can also use keys to move around.

The Tab key also moves the cursor from pane to pane.

- **Tab** and **Shift+Tab** move you between groups (such as the Item Type and Break Before groups). A text box (like Item Text) also counts as a group.
- Once you're in a group, you can use the arrow keys to move around and to select radio buttons. Use the *Spacebar* to check checkboxes. (Note that the radio buttons and the check box under Initial State are different groups.)
- Use **F6** or **Shift+F6** to move between the Outline pane and the dialog box.

Your selections take effect when you do any of the following:

- Press **Enter** to enter the change
- Press **Ins** or choose **Menu | New Menu Item** to enter the change and insert a new item
- Use **Ctrl+↑**, **Ctrl+↓**, or the mouse, to move to another statement in the outline
- Press **Ctrl+P** or choose **Menu | New Pop-up** to insert a new pop-up
- Press **Ctrl+S** or choose **Menu | New Separator** to insert a new separator

The following table describes the selections you can make in the dialog box (the selections are explained in more detail later in the sections on defining menu items and pop-up menus):

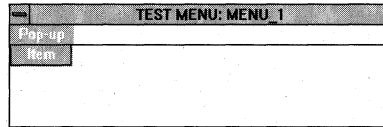
Table 5.1
Menu editor dialog box
selections

Selection	Description
Item Text	The name of a menu command or pop-up command, and an optional description of its accelerator, if it has one.
Item ID	A unique ID for the menu item. This option applies only to menu items and not to pop-up commands or separators.
Item Type	Can be a pop-up menu, a menu item, or a menu separator.
Break Before	Controls the format of menu commands in the menu bar and in pop-up menus. Choose one of the following options:
No Break	There is no break before this command.
Menu Break	Starts a new line in the menu bar or a new column in a pop-up menu.
Menu Bar Break	Starts a new line in the menu bar. In a pop-up menu, starts a new column and draws a vertical line to separate the columns.
Help Break	Moves the menu item to the far right of the menu bar. Use this option only with top-level menu items in the menu bar.
Initial State	The Enabled, Disabled, and Grayed selections in the left column are radio buttons (you can choose only one). The Checked box is a check box and can be selected separately. For example, you can choose both Enabled and Checked.
Enabled	Makes the command work when the user chooses it.
Disabled	Disables the command. The user won't be able to distinguish between Enabled and Disabled commands; they'll look the same on the menu.
Grayed	Disables the command and grays the displayed text. The shading lets the user know the command is not currently available.
Checked	Places a check next to the command. Choose this option if the item will function as a toggle and you want the initial state of the command to be On.

The Test Menu pane

Figure 5.3
A newly-created menu

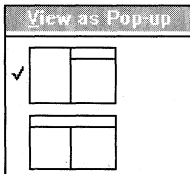
The Test Menu pane, above the Outline window, displays your menu and lets you test it.



As you can see, the pop-up menu for the default Pop-up command contains a single command, Item. The Menu editor automatically updates this test menu as you make changes to the outline.

You can use the View menu to change the way the test menu is displayed and its position relative to the other panes.

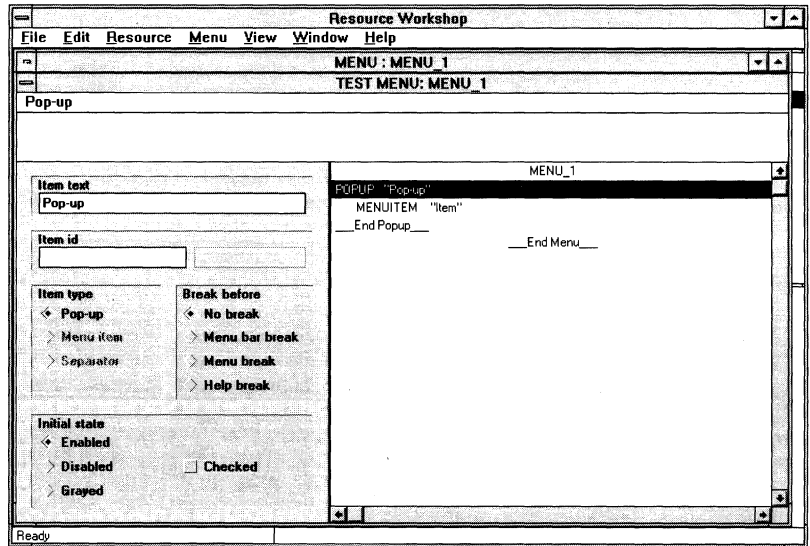
Table 5.2
View menu selections



Menu choice	Description
View as Pop-up	Controls whether the test menu is displayed on the menu bar or as pop-up menu. Leaving this option unchecked (the default) displays the test menu across the menu bar. This option works fine if you're working on an entire menu at once, because the top level of your menu would be displayed across the menu bar in your program. However, if you develop menus as separate resources and the current menu you're working on is a pop-up or floating menu, you'll want to see it as it actually appears. In that case, check this option to display the test menu as a pop-up menu. The command <i>Pop-up</i> appears on the test menu, and you select Pop-up to display the menu itself.
First graphic	This graphic represents the default configuration of the panes, with the test menu over the Outline pane and to the right of the dialog box.
Second graphic	Check this graphic to put the test menu across the top of the edit window, like a normal menu bar.

The following figure shows the Menu editor in the alternate format (with the second graphic checked):

Figure 5.4
The Menu editor window in
alternate format



Starting the Menu editor

Whether you want to create a new menu or edit an existing one, you need the Menu editor. The Menu editor displays a menu you can modify to fit your application. The following sections detail the different ways to start the Menu editor.

Creating a new menu

See Chapter 3 for more information on opening projects.

To create a new menu,

1. Make sure you've already opened the project you want to add the menu to. You can choose File | New Project to create a new project or File | Open Project to open an existing project.
2. Once you've opened a project, choose Resource | New to create a new resource for that project. The New Resource dialog box appears.
3. In the New Resource dialog box, scroll the Resource Type list to MENU, then either double-click MENU or click it and then click OK.

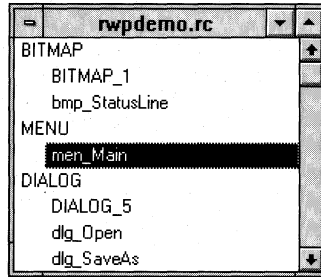
Resource Workshop displays the Menu editor with a default menu in it that you can begin customizing.

Editing an existing menu

To edit an existing menu, open the project in which the menu is stored and do one of the following:

- Double-click the menu name in the Project window.
- Highlight the menu name and choose Resource | Edit.

Figure 5.5
The RWPDEMO Project window



For an example of this screen, see Figure 5.2 on page 117.

Resource Workshop displays the Menu editor with the menu you have chosen loaded into it.

Customizing a menu

Once you have a menu loaded into the Menu editor, you're ready to begin customizing it. Using the Menu editor, you can define new menu commands, pop-up menus, and separators, and you can move, copy, or delete any part of the menu.

Adding a new statement

Whenever you add a new statement to a menu (an item, separator, or pop-up menu), you must first position the cursor in the Outline window on the line preceding where the statement is to go. If you want to insert a statement before the first statement in the outline, move up to the top line, the title line, first. The sections that follow on defining items, pop-up menus, and separators explain in detail how to insert each of these statements.

When you've decided where the new statement is to go and you've highlighted the appropriate line, you can add a new statement by choosing the Menu command.

Figure 5.6
The Menu menu

New menu item	Ins
New separator	Ctrl+S
New file pop-up	
New edit pop-up	
New help pop-up	
Check duplicates	

You can press the Ins key to insert a new command.

Use this menu to insert new menu commands, pop-up menus, and menu separators. You can also insert a complete, standard File, Edit, or Help pop-up menu, and you can test your menu for duplicate IDs (more on this subject later).

Moving and copying

Edit | Cut deletes the selected item, but lets you paste it elsewhere.

You can use Cut, Copy, and Paste on the Edit menu to move and copy the statements in the outline of the Menu editor.

Highlight the statement you want to move or copy, then choose Edit | Cut or Edit | Copy. Next highlight the statement after which you want to insert the copy and choose Paste.

Undoing errors

As with other Resource Workshop editors, the Menu editor lets you undo and redo changes. Choose Edit | Undo or Edit | Redo or use the accelerators *Alt+Backspace* (Undo) and *Shift+Alt+Backspace* (Redo).

For example, if you mistakenly change a menu item into a separator, you can select Edit | Undo. Resource Workshop indicates which change it is going to undo (in this case, it displays "Undo Change Item"). Press *Enter*, click Undo, or press *Alt+BkSp* to undo the change.

Customizing a menu item

The choices in the menu bar are usually pop-up commands.

A menu item (or command) causes the application to initiate an action (as opposed to a pop-up command, which simply brings up a menu). Usually, menu commands are contained in pop-up menus. For example, in most Windows applications, if you choose the File pop-up command, it displays a pop-up menu from which you choose the Open menu command.

To customize a menu item, you must first either create a new one or open an existing one. You can then customize the menu item in the dialog box.

New menu items The first step in creating a new menu item is deciding where to insert it.

Choosing where to insert a new menu item

Before you insert a new menu item, decide where you want it to appear in the menu and highlight the previous line in the Outline pane. In general, if you're putting the new item after another statement, you highlight the existing statement and insert the new one below it.

This rule gets a little more complicated if you want to insert a command on the same level as, but after, a pop-up command, because a pop-up command is followed by a pop-up menu. If you're not careful, the new item will wind up in the pop-up menu rather than on the same level as the original pop-up command.

See page 135 to see how to create the Widgets menu.

For example, let's say you want to add two items to the following menu: a new command called *Stored Order* on the Arrange List pop-up menu, and a *Quit* command that appears on the menu bar next to Widgets.

Figure 5.7
The Widgets menu

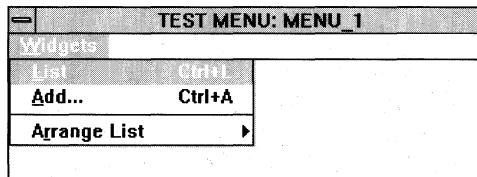
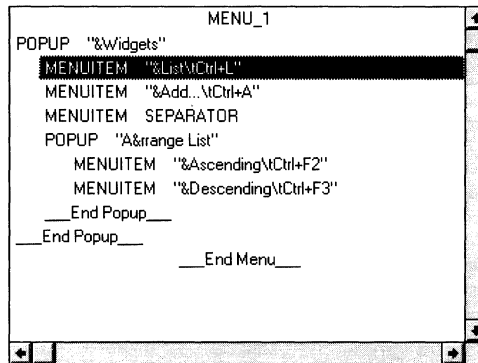


Figure 5.8
Script outline for the Widgets
menu



Adding the Stored Order command is fairly easy. You look for the statement `POPUP "Arrange List..."` and then decide where the new command is to go on the pop-up menu.

- If it's to become the first command, you highlight `POPUP "Arrange List..."` and add the command.
- If it's to follow one of the commands in the pop-up menu (like Descending), you highlight that command (`MENUITEM Descending`) and add the command.

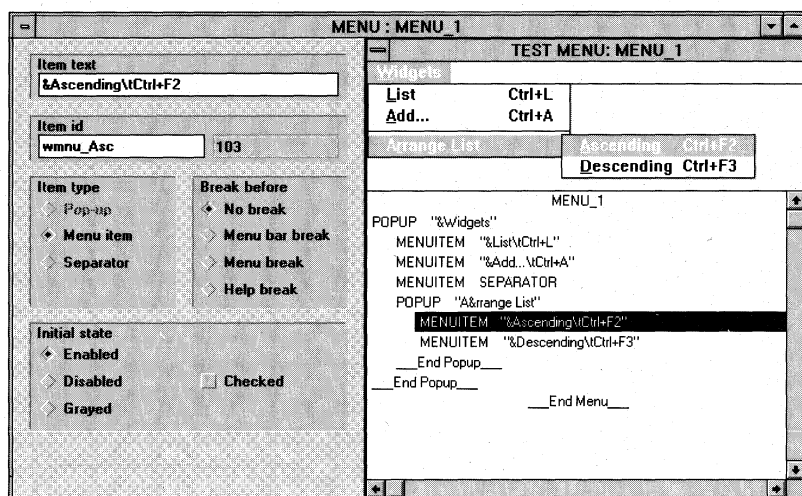
Adding the Quit command is more complicated. If you try to add the command by highlighting Widgets, the new command will appear just below Widgets and will be part of the Widgets pop-up menu. Since you want this command to be on the menu bar with Widgets, you have to add it after the Widgets pop-up menu.

The Menu editor indicates the beginning of a pop-up menu by the statement `POPUP` and the end by the statement `_END POPUP_`. Since the Arrange List pop-up menu is embedded in the Widgets pop-up menu, there are two `_END POPUP_` statements. In the outline, an embedded pop-up menu is indented relative to the pop-up menu it's embedded in, making it easy to see which menu is embedded in another and to find the appropriate ending statement.

If you look at the outline, you see that the `_END POPUP_` statement for the Arrange List menu is indented relative to the `_END POPUP_` statement that follows it. That second statement ends the Widgets pop-up menu.

To add the Quit command, highlight the second `_END POPUP_` statement and add the command. When you're done, the outline and menu look like this:

Figure 5.9
Menu with new items
inserted



Creating the menu item

Now that you know where to insert the menu item, you can create it as follows:

1. Highlight the appropriate line in the Outline pane. Remember, the new menu item is inserted after the highlighted line.
2. Press *Ins* or Choose **Menu | New Menu Item**. The Menu editor creates a new menu item, gives it a default name ("Item") and a default menu ID, and highlights it in the outline.
3. Use the dialog box to the left of the Outline pane to further define the new menu item. (See the "Using the dialog box" section.)

Selecting a menu item To select a menu item for customizing,

1. Highlight the menu item by using the mouse, the *Ctrl*+*↑* and *Ctrl*+*↓* keys, or the arrow keys (if you're in the Outline pane).
2. Edit the fields in the dialog box.

Using the dialog box In the dialog box, use the mouse or the *Tab* or *Shift+Tab* keys to position on the field you want to edit.

Entering item text

The Item Text is the text that displays in the menu.

If you want the user to be able to choose the command from the menu by typing one of the letters in the command, put an ampersand (&) immediately before that letter. Windows will display the command with that letter underlined.

See Chapter 6 for more information on accelerators.

If you plan to link an accelerator key to this command, you can add accelerator text to your menus. For example, if your application includes a *Shift+Del* accelerator that duplicates the Cut command on the Edit menu, you can add the text “Shift+Del” next to that command.

- Use the tab character (\t) to separate the menu title from the accelerator text with a tab (for example, &Cut\tShift+Del).
- Use the right-justify character (\a) to right-justify accelerator text (for example, &Cut\aShift+Del).

Windows applications generally use the plus sign to show key combinations, like *Shift+Del* or *Ctrl+Shift+F4*.

Entering an item ID

All menu items must have a unique numeric ID. Resource Workshop creates a default value for the ID that’s different from all the other menu IDs in this menu. You can type either a unique number or a unique name (an identifier).

See Chapter 3 for more information about identifiers.

If you type a name, Resource Workshop checks to see if an identifier by that name already exists. If one doesn’t exist, you see a prompt asking if you want to create one. If you choose to create a new identifier, Resource Workshop shows you a dialog box that displays the current value it has assigned to the identifier. You can accept this value or type a new number and have Resource Workshop create the identifier for you.

If you’re linking an accelerator key to this command, you need to use this identifier in the Accelerator editor. You might also want

to have the Accelerator editor open so you can switch to that editor and enter accelerators as you create menu items.

The remaining dialog box options

Most of the remaining dialog box options are explained in Table 5.1 on page 119. The one remaining option that applies only to menu items is the Checked option, which puts a check mark next to menu items that toggle off and on. A check mark means the command is On.

Since menu bar commands and pop-up commands don't toggle, you use this option only with menu items on pop-up menus.

For example, on the Menu editor menu bar, the View command displays a pop-up menu with two graphics representing the configuration of the panes. The icon representing the current configuration is checked, meaning that it is On.

The View menu is described on page 120.

Customizing a pop-up command

A *pop-up command* produces a menu that displays additional choices. In most Windows applications, all items in the menu bar across the top of the window are pop-up commands.

To customize a pop-up command, you must either create a new one or open an existing one. You can then customize the pop-up command in the dialog box.

New pop-up commands

The first step in creating a new pop-up command is deciding where to insert it.

Choosing where to insert a new pop-up command

You can insert a pop-up command in the top level of your menu to make it appear on the menu bar. You can also embed pop-up commands in other pop-up menus, as shown in the sample menu in Figure 5.7 on page 124. In that menu, Widget is a top-level pop-up command, so it appears on the menu bar. The Widget pop-up menu includes menu commands as well as the embedded Arrange List pop-up command.

Whether a pop-up command is displayed on the menu bar or embedded in another pop-up menu depends on where you insert

it. For guidance, see “Choosing where to insert a new menu item” on page 124.

Adding a new pop-up command

Once you decide where you want a pop-up command, here are the basic steps for adding it:

1. Highlight the appropriate line in the Outline pane. The new pop-up command is inserted after the highlighted line.
2. Press **Ctrl+P** or choose **Menu | New Pop-up**.
3. Use the dialog box on the left side of the Menu editor to further define the pop-up command. (See “Using the Dialog Box” later in this section.)

Selecting a pop-up command

To select a pop-up command for editing,

1. Highlight the pop-up command by using the mouse, the **Ctrl+↑** and **Ctrl+↓** keys, or the arrow keys (if you’re in the Outline pane).
2. Edit the fields in the dialog box.

Using the dialog box

In the dialog box, use the mouse or the **Tab** or **Shift+Tab** keys to position on the field you want to edit.

Entering item text

The Item Text is the text that displays in the menu.

If you want the user to be able to choose the pop-up command from the menu by typing one of the letters in the command, put an ampersand (&) immediately before that letter. Windows will display the command with that letter underlined.

- Unlike menu items, pop-ups don’t have accelerator keys. You’d be more likely to tie an accelerator to a command on the pop-up menu than to the pop-up command itself.

The remaining dialog box fields

See Table 5.1 on page 119 for a description of the dialog box.

You don’t need to set the Item ID and Item Type options for pop-up commands, but you might want to set the Break Before options and the Initial State options (Enabled, Disabled, and

Grayed, but not Checked—Checked applies only to menu commands, not pop-ups).

Defining a menu separator

A *menu separator* is a line that separates different groups of choices in a pop-up menu. Since it's only a line in the menu, you won't customize it, but you will decide where it goes and then create it.

You can insert a menu separator anywhere in a pop-up menu. In general, you use menu separators to group related choices. For guidance on how to insert new choices into your menu, see "Choosing where to insert a new menu item" on page 124.

Once you decide where you want a separator,

1. Highlight the appropriate line in the outline area of the Menu editor. The new separator is inserted after the highlighted line.
2. Press *Ctrl+S* or choose *Menu | New Separator*.

You don't have to set any of the options in the dialog box on the left side of the Menu editor.

Deleting a menu statement

Highlight the statement you want to delete, then press *Del* or choose *Edit | Delete* to delete it, or choose *Edit | Cut* to delete the statement and copy it to the clipboard.

You can't delete *END POPUP* statements. If you want to delete a pop-up command and all the items contained in a pop-up menu, delete the starting *POPUP* statement.

For example, suppose a pop-up menu contains four items: two menu commands, a menu separator, and a pop-up command. You can delete any three of these items from the pop-up menu, but you can't delete all four without deleting the entire pop-up menu. A pop-up menu must always contain at least one menu item, another pop-up command, or a menu separator.

Testing a menu

The Menu editor gives you immediate testing capability. The test menu is updated as you make changes to your menu, so you can test changes as often as you want by using the Test Menu pane.

For example, if you add a new menu command to a pop-up menu, click the new command in the test menu to see how your new command looks.

The Menu editor also has a built-in debugging tool that you can use to test for duplicate menu item IDs. If you choose Menu | Check Duplicates, the Menu editor searches for duplicates and, if it finds any, displays a dialog box with the message “Duplicate command value found.”

When you close this message box, the Menu editor highlights the statement that contains the duplicate value. What you do next depends on whether the menu ID is an identifier (with a name) or simply a numeric value. To see which it is, look at the Item ID text box in the Dialog Box pane.

- If the menu ID is a number, enter a new number that doesn’t conflict with the other menu IDs.
- If the menu ID is an identifier, you have to bring up the Identifiers dialog box to change its value, as follows:
 1. Choose Resource | Identifiers to display the Identifiers dialog box.
 2. Scroll down the list of identifiers until you find the one you want.
 3. Press the Change button and type a new value that doesn’t conflict with the other menu IDs.
 4. Click OK or press *Enter* to change the value.
 5. Click in the Menu editor window to continue editing your menu. (You can leave the Identifiers dialog box open for later use.)

For example, if, in the sample menu on page 135, you had assigned the value 101 to two identifiers, *wmnu_List* and *wmnu_Asc*, Menu | Check Duplicates would produce the message “Duplicate command value found.” Since *wmnu_Asc* is the second of the two identifiers in the menu, the Menu editor would highlight *wmnu_Asc*. You could then bring up the Identifiers

dialog box by using Resource | Identifiers, highlight *wmnu_Asc*, and change its value to something other than 101, 102, or 104 (the values of the other identifiers in the menu).

Saving changes

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. There are two primary options you can use to save changes you've made to a menu resource:

- Saving the entire project
- Saving the menu resource as a file

Saving the project

This is the Save option you'll use most often.

To save the entire project, choose File | Save Project. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. Any changed resource linked to an external file is updated in the external file.

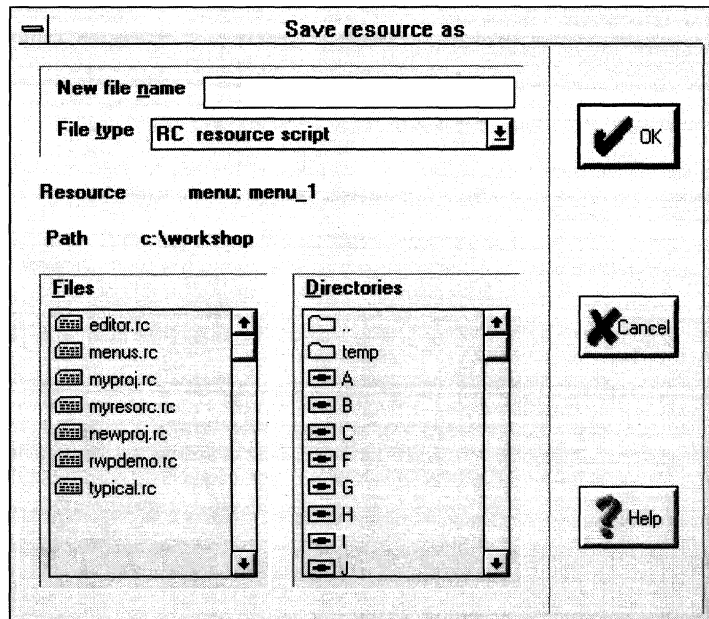
Saving the menu resource as a file

Saving the entire project also saves the menu resource you're working on. However, if you want to save only the menu resource, you can choose Resource | Save Resource As to save the resource as a file. Because this step is more complicated than just saving the project, you aren't likely to do it unless you want to put the menu resource in a separate file for the first time.

To save the resource as a file,

1. Choose Resource | Save Resource As. Resource Workshop displays the Save Resource As dialog.

Figure 5.10
The Save Resource As dialog
box for a menu resource



2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or enter the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or click OK.

Editing a menu resource script

To work with the resource script of a menu, select the menu name from the project window by clicking it, then choose Resource | Edit As Text to display the resource script in the internal text editor.

See page 41 for a description of the internal text editor.

For example, to edit the resource script for the sample menu at the end of this chapter, you can open the project containing that menu, then highlight the menu and choose Resource | Edit As Text. Resource Workshop opens its internal text editor and displays the source code as follows:

See the online Help index for a description of the resource script options for menus.

```
MENU_1 MENU
BEGIN
  POPUP "&Widgets"
  BEGIN
    MENUITEM "&List\tCtrl+L", wmenu_List
    MENUITEM "&Add...\tCtrl+A", wmenu_Add
    MENUITEM SEPARATOR
    POPUP "A&rrange List"
    BEGIN
      MENUITEM "&Ascending\tCtrl+F2", wmenu_Asc
      MENUITEM "&Descending\tCtrl+F3", wmenu_Desc
    END
  END
END
```

You can then use the editor to make changes to the source code directly. For example, to change the menu memory options from Moveable, Discardable, and Pure (the defaults) to Fixed, Nondiscardable, and Pure,

You can also make these changes by choosing Resource | Memory Options.

1. In the text editor, alter the first line of the script to read as follows:

```
MENU_1 MENU FIXED NONDISCARDABLE PURE
```
2. Double-click the edit window's Control-menu box to close the window.
(If you want to stay in the menu editor but compile what you just entered to see if it is correct, you can choose Compile | Compile Now instead of closing the editor.)
3. Resource Workshop asks you if you want to compile. When you choose OK, Resource Workshop compiles the menu and exits to the project window.

■■■■► *Don't spend any time inserting comments in your resource script or formatting the text, because the Resource Workshop incremental compiler does its own formatting and discards all comments.*

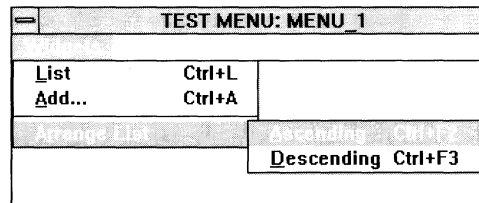
A sample menu

This section takes you through the creation of a very simple menu. All the basics of how to use the Menu editor are explained here.

Suppose you want to create a Widget pop-up menu with three commands. The first two commands let users look at existing widgets (List) or add new widgets (Add). The third command, Arrange List, produces a pop-up menu with two additional commands (Ascending and Descending) that let users choose the sorted order of the list of widgets.

Here's the menu design:

Figure 5.11
The sample menu



In this menu, Widget and Arrange List are pop-up commands. When a user clicks Widget, a menu pops up to display the commands List, Add, and Arrange List. When the user clicks on Arrange List, another menu pops up to display Ascending and Descending.

List and Add are menu items (or commands). When the user clicks either of these commands, the application performs an action. The List command produces a list of widgets, and the Add command displays a dialog box that prompts the user for a widget to add to the list.

Notice the menu separator that groups the List and Add commands separately from the Arrange List pop-up command.

Creating a menu using a text editor

Without the Menu editor, you would have to produce two files to create this sample menu, one file containing definitions of the identifiers and another containing the menu file itself.

The identifier file (called WDGCONST.PAS) would contain the following code for a Pascal program:

```
unit RWPDemoC;
interface
const
    wmnu_List = 101;
    wmnu_Add  = 102;
    wmnu_Asc  = 103;
    wmnu_Desc = 104;
implementation
end.
```

The file containing the menu would contain the following code:

```
#include "wdgconst.pas"

MENU_1 MENU
BEGIN
    POPUP "&Widgets"
    BEGIN
        MENUITEM "&List\tCtrl+L", wmnu_List
        MENUITEM "&Add...\tCtrl+A", wmnu_Add
        MENUITEM SEPARATOR
        POPUP "A&rrange List"
        BEGIN
            MENUITEM "&Ascending\tCtrl+F2", wmnu_Asc
            MENUITEM "&Descending\tCtrl+F3", wmnu_Desc
        END
    END
END
```

Creating a menu using the Menu editor

Not only is it easier to create the sample menu using the Menu editor, but you can test the menu as you create it. Other advantages of the Menu editor are that it handles menu IDs for you, and it even stores identifiers in a separate identifier file if you tell it to.

Creating the menu To create the sample menu,

1. Make sure you've already opened a project. If you're doing the sample resources in each chapter, open MYPROJ.RC. (See Chapter 3 if you need information about opening a project.)
2. With a project open, choose Resource | New.

3. Resource Workshop displays the New Resource dialog box. Scroll the Resource Type list until you see MENU, click on it, then click OK.

Resource Workshop adds a new menu resource to the project window, then displays the new menu in the Menu editor with the first statement in the outline, POPUP "Pop-up", highlighted.

4. To rename the initial menu statement from Pop-up to Widgets, type `&Widgets` in the dialog box in the Item Text text box, then press *Enter*.

The Menu editor updates both the test menu and the source code.



Notice that the *W* in Widgets is underlined, indicating that when this menu is displayed, you can press *W* to choose the Widgets command.

Adding commands to
the Widgets menu

Next, add the commands to the menu popped up by the Widgets command.

1. To rename the first menu item and add text indicating the accelerator key, press *Ctrl+↓* to highlight the second line of the outline (MENUITEM "Item"), then type `&List\tCtrl+L` in the Item Text text box.

2. Tab to the Item ID text box and type `wmnu_List` to enter an identifier for this command (to be used later in the Accelerator editor to create the *Ctrl+L* accelerator for the command).

Resource Workshop asks you if you want to create a new identifier. Press *Enter* to do so, then in the New Identifier dialog box press *Enter* to accept the current value displayed.

At this point, you'd normally open the Accelerator editor and create the *Ctrl+L* accelerator key. It's usually a good idea to create accelerators as you create the menu, because you need the identifiers for the menu commands to create the associated accelerators. However, since we haven't discussed accelerators yet, we'll defer this part of the demo to page 151 in the next chapter.

3. With the second line of the outline (the List command) highlighted, add a new menu command either by pressing *Ins* or by choosing Menu | New Menu Item.

4. Type `&Add...\tCtrl+A` in the Item Text box to change the text for the new menu item. (The ellipsis (...) after the Add command indicates that the command brings up a dialog box.)
5. Since you've indicated that `Ctrl+A` is to be the accelerator for this menu item, create an identifier for the item by tabbing to the Item ID field, typing `wmnu_Add`, pressing `Enter`, and responding to the prompts as before for the `wmnu_List` identifier.
6. With the third line of the outline (the Add command) highlighted, press `Ctrl+P`, then change the text to `A&rrange List...` to add the Arrange List pop-up command to the menu. Because you want two options to appear when the user chooses Arrange List, you need to define Arrange List as a pop-up command rather than as another menu item.
7. Highlight the line defining the Add command, then press `Ctrl+S` to put a separator after this command.

Adding commands to the Arrange List menu

To define the menu commands Ascending and Descending in the Arrange List pop-up menu,

1. Change the menu item "Item" to "&Ascending" (this is the item Resource Workshop created when you created the Arrange List menu).
2. Press `Ins` to add a new menu item after "&Ascending", then rename it "&Descending".
3. If you wish, you can indicate accelerators for each of these commands as you did for the List and Add commands. If you do indicate accelerators, also define identifiers for the commands.

Testing the menu

Test the menu by clicking on the Widgets command in the Test Menu pane and dragging down to the Arrange List command. Your menu should look like Figure 5.11 on page 135.

You can also test for duplicate values in menu IDs by choosing `Menu | Check Duplicates`.

- If there are no duplicates, you get the message "No duplicates found."
- If there are duplicates, you get the message "Duplicate command value found." You can then go through the steps on page 131 to correct the duplicate values.

Creating accelerators

An *accelerator* is a hot key, a key combination the user presses to perform a task with your application. It substitutes for a menu command and, just like a menu command, creates a WM_COMMAND or WM_SYSCOMMAND message that tells the application what to do next.

See Chapter 5 for more information about using accelerators with menu commands.

Usually, you create accelerators to duplicate commands on pop-up menus. For example, if you open the Edit menu in most Windows applications, you see these accelerators: *Shift+Del* (the Cut command), *Ctrl+Ins* (the Copy command), and *Shift+Ins* (the Paste command).

Figure 6.1
An Edit menu with
accelerators

Undo	Alt+Backspace
Cut	Shift+Del
Copy	Ctrl+Ins
Paste	Shift+Ins
Clear Delete	

You store accelerator definitions in an accelerator table (the accelerator resource). Each entry in the table is an accelerator that defines the key combination a user must press and the command it produces. If you like, you can create multiple accelerator tables (or resources) for different parts of your menu.

Resource Workshop provides you with an Accelerator editor that makes it easy to create and edit accelerators for your application. When working with accelerators, you perform four primary tasks:

- Starting the Menu editor
- Starting the Accelerator editor
- Customizing an accelerator table and testing it for duplicate keys
- Saving the accelerator table

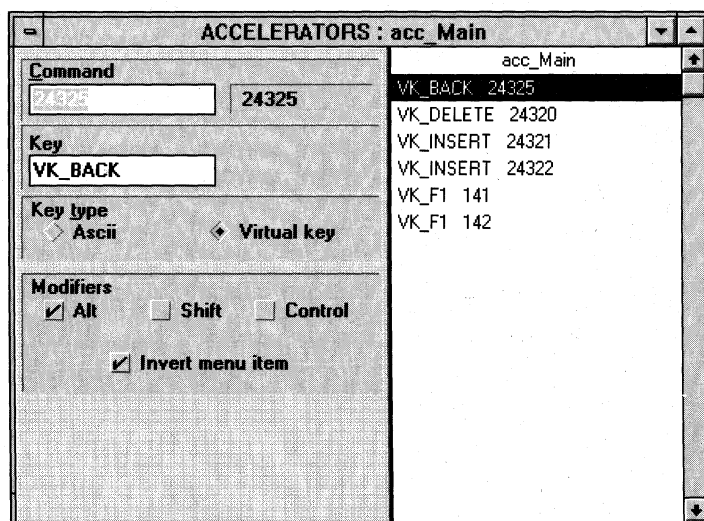
See page 151 for an example illustrating how to use the Accelerator editor.

The first task, starting the Menu editor, displays the menu associated with the accelerators you're working on. The second task, starting the Accelerator editor, displays an accelerator table ready for you to customize. The third task, customizing the accelerator and testing it, is done in the Accelerator editor itself. The fourth task, saving the accelerator, happens automatically when you save your project.

Before discussing these tasks, we show you how the Accelerator editor screen and features are organized. The next sections cover the four primary tasks you perform in creating an accelerator, followed by a description of how to check for duplicate key combinations, a description of how to edit an Accelerator resource script, and an example of how to create an accelerator table.

If you open the demo project RWPDEMO.RC and double-click on the Accelerator resource, Resource Workshop displays the following accelerator table:

Figure 6.2
RWPDEMO accelerator table
in Accelerator editor



Using the Accelerator editor

The Accelerator editor screen is divided into two panes, the Outline pane and the dialog box (see Figure 6.2).

You can move between these two panes by using the mouse or the *F6* and *Shift+F6* keys.

The Outline pane

The pane on the right, the Outline pane, shows you all the accelerators defined in the table. The top line in this pane is the name of the accelerator table. The lines below it are accelerator entries.

An accelerator entry has two parts:

■ The first part is the accelerator key.

- If the key is a virtual key (an identifier in uppercase letters that starts with *VK_* and represents keys like function keys and arrow keys), you can look at the Modifiers check boxes in the dialog box to see if the key is combined with *Ctrl*, *Alt*, *Shift*, or any combination of the three.

For example, *Ctrl+Alt+Shift+F1* is represented in the Outline pane as *VK_F1*. In addition, the *Alt*, *Shift*, and *Control* check boxes are checked in the dialog box.

- If the key is an ASCII key (a displayable key surrounded by quotation marks), it can be an uppercase or lowercase letter, and it can be preceded by a carat (^) to indicate that it is combined with *Ctrl* (*Ctrl+W* is represented as "*^W*"). You can look at the *Alt* check box in the dialog box to see if the key is combined with *Alt*.

For example, *Ctrl+Alt+W* is represented in the Outline pane as "*^W*". In addition, the *Alt* check box is checked in the dialog box.

■ The second part is the item ID of the command to which the accelerator is tied. This ID is either an integer or the name of an identifier.

When you're in this pane, you can select an accelerator by using the mouse, the scroll bar, or the arrow keys. You needn't be in this pane to select an accelerator; from the dialog box, you can use the mouse or press *Ctrl+↑* or *Ctrl+↓*.

See page 146 for an explanation of ASCII and virtual keys.

The Dialog Box pane

Selecting one of the accelerators in the Outline pane shows its settings in the dialog box on the left. With an accelerator selected, you can make changes to it in the dialog box, such as entering a new key combination or associating the accelerator with another command.

To move around inside the dialog box, you can use the mouse to position anywhere and make selections. You can also use the following keys to move around:

- *Tab* and *Shift+Tab* move you between groups (such as the Key Type and Modifiers groups). A text entry box (like Key) and a check box (like Alt) also counts as a group.



If you press *Tab* to move from the Command text box to the Key text box, the Accelerator editor changes to Key Value mode, in which you can press any key to enter it as an accelerator. To exit from this mode, click the mouse or press *Alt+Esc*.

- Once you're in a group, you can use the arrow keys to move around and to select radio buttons. Use the *Spacebar* to check check boxes.

Your selections take effect when you press *Enter* (to change the accelerator) or *Ins* (to create a new accelerator), or you move to another accelerator in the outline.

The following table describes the selections you can make in the dialog box. These selections are described in more detail later in the chapter.

Table 6.1
Accelerator editor dialog
box selections

Selection	Description
Command	The item ID for the command the accelerator is to execute. You can enter either an integer or an identifier.
Key	The accelerator key. You can enter the key in Manual mode (make your own decisions about whether the key is ASCII—surrounded with quotation marks, or virtual—an identifier or integer) or in Key Value mode (if you press a key, the editor decides if it's an ASCII or virtual key and enters it for you in the appropriate format). If you tab into this text box from the Command field, you're automatically in Key Value mode.

Table 6.1: Accelerator editor dialog box selections (continued)

Selection	Description
Key Type	Either ASCII or virtual. If you're in Key Value mode, the Accelerator editor sets these radio buttons automatically.
ASCII	The key you're entering is a displayable key and is represented using ASCII conventions (the character is surrounded by quotation marks and the <i>Ctrl</i> part is represented as a carat— <i>Ctrl</i> + <i>L</i> appears as “^L”).
Virtual Key	The key you're entering does not have a standard character representation and must be entered as an identifier (for example, the <i>F1</i> key is represented by the Windows identifier <i>VK_F1</i>).
Modifiers	A set of check boxes that define key combinations and enable or disable flashing the associated main menu item. The descriptions that follow tell what the option means if it's checked.
Alt	The accelerator is a key combination that includes the <i>Alt</i> key (for example, <i>Alt+W</i>).
Shift	The accelerator is a key combination that includes the <i>Shift</i> key (for example, <i>Shift+F1</i>).
Ctrl	The accelerator is a key combination that includes the <i>Ctrl</i> key (for example, <i>Ctrl+W</i>).
Invert Menu Item	Using the accelerator causes the associated menu bar command to flash (to invert momentarily).

Starting the menu editor

See Chapter 5 to see how to start the Menu editor.

When working with accelerators, it's usually a good idea to start the menu editor and load in the menu containing the associated commands. That way you can see the command text and the item IDs you need for defining the accelerators. As explained later in this chapter, each accelerator must have an identifier that corresponds to a command on the menu.

To let users know what accelerators they can use, you can add accelerator text to your menus. For example, if your application includes a *Shift+Del* accelerator that duplicates the Cut command on the Edit menu, you can add the text “Shift+Del” next to that command.

Use the tab character (\t) to separate the menu title from the accelerator text (for example, Cut\tShift+Del).



Windows applications by convention use the plus sign to show key combinations, like *Shift+Del* or *Ctrl+Shift+F4*.

Starting the Accelerator editor

Whether you want to create a new accelerator table or edit an existing one, you need to start the Accelerator editor. When the Accelerator editor starts, it displays an accelerator table you can edit to fit the needs of your application. How you start the Accelerator editor depends on whether you want to create a new accelerator table or edit an existing one.

Creating a new accelerator table

See Chapter 3 for more information on opening projects.

To create a new accelerator table,

1. Make sure you've already opened the project you want to add the accelerator table to. You can choose File | New Project to create a new project or File | Open Project to open an existing project.
2. Once you've opened a project, choose Resource | New to create a new resource for that project. The New Resource dialog box appears.
3. In the New Resource dialog box, scroll the Resource Type list to ACCELERATOR, then either double-click ACCELERATOR or click it and then click OK.

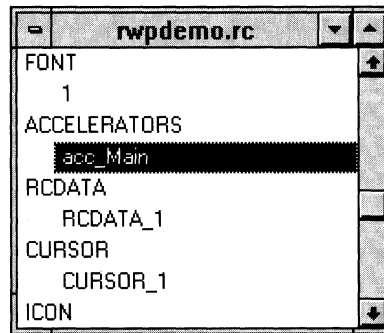
Resource Workshop displays the Accelerator editor with a default accelerator table in it that you can begin customizing.

Editing an existing accelerator table

To edit an existing accelerator resource, open the project in which the accelerator resource is stored and do one of the following:

- Double-click the accelerator resource name in the Project window.
- Highlight the accelerator resource name and choose Resource | Edit.

Figure 6.3
RWPDEMO project window



Resource Workshop displays the Accelerator editor with the accelerator table you have chosen loaded into it (see Figure 6.2).

Customizing an accelerator table

Once you have an accelerator table loaded into the Accelerator editor, you're ready to begin customizing it. Using the Accelerator editor, you can define and change accelerators, and you can specify an accelerator key combination simply by pressing the desired key combination. You can also copy or delete any accelerators in the table, and you can test the table for duplicate identifier values.

Selecting an accelerator key

To customize an accelerator, you can either create a new one or select one that already exists.

- To add a new accelerator to the accelerator table, press *Ins* or Choose Accelerator | New Item. The new key appears in the outline below the currently selected line with the default values of 0 (zero) for the key value and a unique integer value for the command ID. (The Accelerator editor checks the command values for the other accelerators in the table to ensure that the new value is different.)
- To select a currently existing accelerator, use the mouse or press *Ctrl+↑* or *Ctrl+↓* to highlight the accelerator in the Outline pane. The Accelerator editor displays the current values of the accelerator in the dialog box.

Using the dialog box

Once you've selected an accelerator for editing, you can use the dialog box to customize it.

Setting the command value

Under Command, type the item ID for the command the accelerator is to execute. You can enter either an integer or an identifier.

If you have the menu loaded into the Menu editor, you can change windows to the Menu editor and select the appropriate menu command. Note the command's item ID and use that same ID in the Command text box for the accelerator.

You can use either the integer value of the ID or its identifier name.

If you enter an identifier name and you see the message "Create a new identifier: *identifier name*?", you've probably mistyped the name. Click No and check your spelling.

If you deliberately enter a new identifier because you intend to add the associated menu item to the menu later, a prompt appears asking you if you want to create a new identifier. (If you don't get this prompt, the identifier already exists, and you might not want to use it.) Since you're creating a new identifier, click Yes or press *Enter*, and then enter the identifier's value in the New Identifier dialog box. Be sure to enter a value that's different from the other item IDs in the associated menu.

Understanding ASCII and Virtual keys

You enter a representation of the key itself in the Key text box. However, before describing how to enter keys, it might be helpful to explain what ASCII keys and virtual keys are.

ASCII keys

When you press an ASCII key, it displays on the screen as a character, such as *a*, *N*, *5*, or *%*.

An ASCII key appears in the Key text box surrounded by quotation marks and follows standard conventions for representing ASCII characters. For example, the unshifted *F* key is "*f*", the shifted *F* key is "*F*", and *Ctrl+F* is "*^F*".

Because there is no convention for representing an ASCII key in combination with *Alt*, the Accelerator editor provides you with an *Alt* checkbox to indicate this key combination.

- Typically, you don't use single ASCII characters as accelerator keys; instead, you combine them with *Alt* or *Ctrl* (you'd use *Ctrl+L* or *Alt+L* rather than *L* alone). If you do use a single ASCII character as an accelerator, you should see Charles Petzold's *Programming Windows* for a discussion of how Windows interprets shifted and unshifted ASCII characters used as accelerators.

See page 6 for the bibliography listing for this book.

Virtual keys

When you press a virtual key, such as *F3* or $\uparrow$, characters might display on the screen, but there's no standard specifying which characters, if any, appear. So what do you enter in the Key text box? What you use are standard Windows identifiers that stand for certain keys, such as *VK_BACK* for *BkSp* and *VK_F1* for *F1*. These identifiers are all defined in *WINDOWS.H*.

- You needn't look up any of these virtual key identifiers if you use Key Value mode (explained later) to insert the key, because the Accelerator editor looks up the correct value and inserts it for you.

Virtual keys themselves have no provision for *Ctrl*, *Shift*, and *Alt* combinations. To represent one of these combinations, you check the appropriate check boxes under *Modifiers*.

Specifying the accelerator key

To specify the accelerator key combination, enter the key combination in the Key text box.

- Your accelerator should be consistent with the accelerators in other Windows applications, so don't use any key combinations required by Windows (such as *Ctrl+Esc*). For guidelines about choosing appropriate key combinations, see IBM's *SAA Common User Access Advanced Interface Design Guide*.

See page 6 for the bibliography listing for this book.

There are two modes you can use to enter the key, manual and Key Value.

Manual mode

If you use the mouse to position on the Key text box or if you press *Alt+Esc* to exit from Key Value mode, you're in Manual mode.

In Manual mode, you yourself specify all the information needed to represent the key. You must decide if the key is an ASCII key or a virtual key and enter it in the correct format. If it's a virtual key, you have to know the correct Windows identifier and type it in uppercase letters. You must also check the appropriate Key Type check box (ASCII or Virtual Key) for the key.

If you want to use *Alt* in combination with an ASCII key, you must check the Alt check box. To use any combination of *Alt*, *Shift*, and *Ctrl* with a virtual key, you check one or more of the check boxes corresponding to these modifiers.

For example, to specify *Alt+F3* as an accelerator key, you type `VK_F3` in the Key text box (be sure to use uppercase letters) and check the Virtual Key and Alt check boxes.

To specify *Alt+Ctrl+F* as an accelerator key, type `^F` or `^f` (*Ctrl* and *Alt* key combinations aren't case sensitive) and check the ASCII and Alt check boxes.

Key Value mode

You get into Key Value mode by tabbing into the Key text box or choosing Accelerator | Key Value. You know you're in this mode when you see the Outline window turn gray and the following message appear in this window: "Use the keyboard to select a key. Click the mouse or press *Alt+Esc* when done."

In Key Value mode, the Accelerator editor does most of your work for you. Any key or key combination you press gets entered in the Key text box as the accelerator. The Accelerator editor determines if the key is an ASCII key or a virtual key and checks the correct Key Type check box. If you press a key combination, the Accelerator editor also checks the appropriate Modifiers check boxes.

For example, if you press *Alt+F3* in Virtual Key mode, the Accelerator editor enters `VK_F3` in the Key text box and checks the Virtual Key and Alt check boxes.

If you press *Alt+Ctrl+F* in Virtual Key mode, the Accelerator editor enters "*^F*" in the Key text box and checks the ASCII and Alt check boxes.

The flash feature Windows has a built-in function that *flashes* a menu-bar command when the user presses an accelerator key for a command associated with the menu-bar command.

For example, if you're using the Windows Write application and you have a block of text selected, pressing *Shift+Ins* (the equivalent of choosing Edit | Insert) causes Windows to temporarily invert (flash) the Edit command on the menu bar. This feature lets the user know which menu the accelerator key is on.

The flash feature (also called *invert menu item*) is on by default when you create an accelerator. You can disable this feature by unchecking the Invert Menu Item option in the Accelerator editor dialog box.

Moving and copying an accelerator

Use the Cut, Copy, and Paste commands in the Edit menu to move and copy accelerators. In the outline, highlight the accelerator you want to move or copy, then choose Edit | Cut (to move it) or Edit | Copy (to copy it). Then highlight the place in the outline after which you want to insert the accelerator and choose Edit | Paste.

Deleting an accelerator

Either use the Edit | Cut or Edit | Delete command to delete accelerators, or use the *Del* key. Highlight the accelerator you want to delete, then press *Del* or choose Edit | Cut or Edit | Delete.

Undoing and redoing changes

As with other Resource Workshop editors, the Accelerator editor lets you undo and redo changes. Choose Edit | Undo or Edit | Redo, or use the accelerators *Alt+Backspace* (Undo) and *Shift+Alt+Backspace* (Redo).

Checking for duplicate key combinations

To ensure that you don't use the same key combination more than once, Resource Workshop lets you debug an accelerator table by searching for duplicate key combinations, as follows:

1. With an accelerator table open, choose **Accelerator | Check Dup Keys**.
2. If two accelerators use the same key combination, the Accelerator editor displays the message "Duplicate key value found" and highlights the second accelerator. Make your changes and continue debugging your accelerator table with **Check Dup Keys** until you see the message "No duplicate key values found."

Editing a resource script for an accelerator table

The Resource Workshop Accelerator editor makes it easy to create and modify accelerators. You can also work with the resource script.

See page 41 for a description of the internal text editor.

If you'd rather work with the resource script of an accelerator table, select the accelerator table from the project window by clicking it, then choose **Resource | Edit As Text**.

Resource Workshop brings up the source script for the resource in its internal text editor.

Here's the resource script example from the end of this chapter:

See the online Help for a description of the resource script options for accelerators.

```
My_Accelerators ACCELERATORS
BEGIN
    "^L", wmnv_List
    "^A", wmnv_Add
    VK_F2, wmnv_Asc , VIRTKEY, CTRL
    VK_F3, wmnv_Desc, VIRTKEY, CTRL
END
```

If you want to keep the Widgets command from flashing when you press **Ctrl+L**, you could use the text editor to add the **NOINVERT** command to the first line of the preceding resource script. (In the Accelerator editor, unchecking the **Invert Menu Item** selection for this accelerator has the same effect.) When you're done, that line reads as follows:

```
"^L", wmenu_List, NOINVERT
```

To save your changes,

1. Choose the Close command from the Editor's System menu.
2. In response to the prompt "Resource has changed. Compile?", click Yes.

Resource Workshop compiles your changes for you and saves them. If there's a syntax error, Resource Workshop puts you back in the text editor so you can correct the error.

■ ■ ■ ■ ■ *Don't spend any time inserting comments in your resource script or formatting the text, because the Resource Workshop incremental compiler does its own formatting and discards all comments.*

Creating a sample accelerator table

Suppose you want to create accelerators for the List, Add, Ascending, and Descending commands in the Widgets menu on page 135. Without Resource Workshop, you'd produce the following resource script to create these accelerators:

```
My_Accelerators ACCELERATORS
BEGIN
    "^L", wmenu_List
    "^A", wmenu_Add
    VK_F2, wmenu_Asc , VIRTKEY, CTRL
    VK_F3, wmenu_Desc, VIRTKEY, CTRL
END
```

With Resource Workshop's Accelerator editor, you do the following:

1. Open MYPROJ.RC. (See Chapter 3 if you need information about opening a project.)
2. Choose Resource | New.
3. In the New Resource dialog box, scroll to the ACCELERATOR resource type and double-click on it to tell Resource Workshop to create a new accelerator table. You see the Accelerator editor with one new entry in it.
4. To open the Menu editor, double-click in the Project window on the Menu resource for the menu you created in Chapter 5.

5. Resize the windows for the Menu editor and the Accelerator editor so you can see both at the same time.
6. In the Menu editor Outline window, highlight the List menu item and note its item ID (*wmnu_List*) and the accelerator you intended for it (*Ctrl+L*).
7. Click on the new accelerator in the Accelerator editor.
8. In the Command text box, enter the name of the identifier you see in the Menu editor's Item ID text box (*wmnu_List*).
9. Tab to the Key text box.
10. Notice that the Outline window turns gray and displays a message indicating that you are now in Key Value mode. Press *Ctrl+L* and notice how the Accelerator editor enters the ASCII value "*^L*" and checks the ASCII check box for you.
11. Press *Alt+Esc* to exit from Key Value mode, then press *Enter* to cause these settings to take effect on the highlighted accelerator key in the Outline pane. Notice that the identifier you entered in the Command text box takes on the same value it has in the menu.
12. Press *Ins* to create a new accelerator.
13. Click on the Menu editor, select the next command with an accelerator key, and note its item ID and accelerator key.
14. Repeat steps 7 through 13, substituting the appropriate identifier and key combination, until all the accelerators are defined.

When you've defined all the accelerators for the menu, you're done. However, for demonstration purposes, we're going to do a few more things.

1. Highlight the last accelerator you created, then press *Del* to delete it (you'll see later how to undo this deletion).
2. Highlight the second accelerator and change its key value to *Ctrl+L*.
3. Choose Accelerator | Check Dup Keys.
4. When the "Duplicate Key Value" message comes up, press *Enter* to clear it.
5. At this point, the Accelerator editor positions you on the duplicate key you just created (the second entry). You can either tab to the Key text box and change its value back, or you can press *Alt+BkSp* to undo the change you made.

6. Choose Accelerator | Check Dup Keys again. You should get the message, "No duplicate keys found."
7. Select Edit. Notice that there's a choice, Undo Delete Item, that refers to your deletion of the last accelerator, and there's another choice, Redo Change Item, that allows you to undo your last undo (change the second accelerator key back to Ctrl+L).
8. Choose Undo Delete Item to restore the accelerator you deleted in step 1.
9. Now if you select Edit, you see that the redo choice has changed to Redo Delete Item. You could trace back and forth through all your accelerator key changes with undo and redo, limited only by the number of Undo Levels you set in the Preferences dialog (File | Preferences).

This example shows you how easy it is to create accelerators by using both the Menu editor and the Accelerator editor. You can switch back and forth between the two editors to see which accelerator keys are associated with which menu commands, and you can use Key Value mode to enter the keys, letting the Accelerator editor do most of your work for you. When you're done, you can check to ensure that you haven't created any duplicate keys. If you have, it's easy to change them, both in the menu and in the accelerator table.

Creating a string table

A *string table* is a table that holds error messages, prompts, or any other text strings your application needs to display. You can store multiple string tables in your project file. Typically, you'll define a separate string table for each logical grouping of your program. For example, you might have strings associated with all the menus called up by the commands on the File menu.

Defining strings of text as a separate resource makes it easy to edit the text without changing your source code. For example, if you need to translate a Windows application into a foreign language, putting most of your text in string tables simplifies the process. (You would still need to translate text in other resources, such as dialog boxes.)

When working with string tables, you perform four primary tasks:

- Starting the String editor
- Creating and editing string tables
- Saving a string table
- Testing a string table

At the end of this chapter you'll find a short tutorial that takes you through the steps of creating and editing a string table. The tutorial begins on page 162.

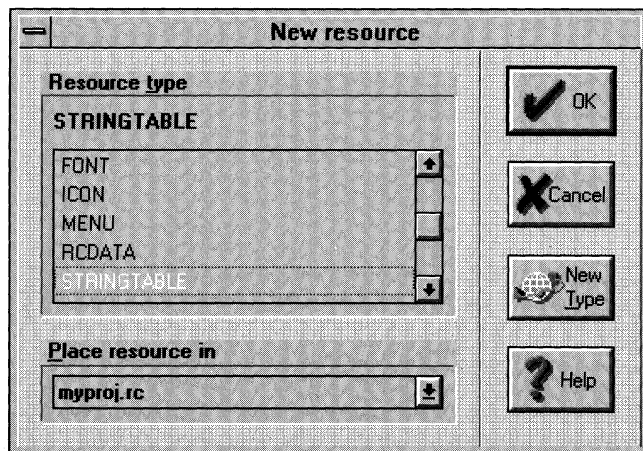
Starting the String editor

How you start the String editor depends on whether you want to create a new string table or edit an existing one.

To start the String editor to create a new string table,

1. Open the project you want to add the string table to.
2. Choose Resource | New.
3. In the New Resource dialog box, select STRINGTABLE in the Resource Type list box and choose OK.

Figure 7.1
The New Resource dialog
box with STRINGTABLE
highlighted



Resource Workshop opens the String editor and places a reference to the new string table in your Project window.

To start the String editor to edit an existing string table,

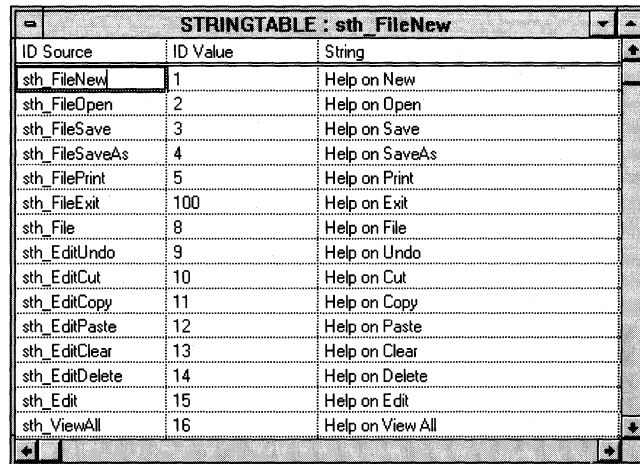
1. Open the project containing the string table you want to edit.
2. Find the string table in the Project window.
3. Double-click the string table entry or select it and choose Resource | Edit.

The string table you selected appears in the String editor.

Working with string tables

When you open the String editor, a string table appears. If you just created a new string table, you won't see any string entries because you haven't entered any yet. If you're editing an existing string table, you'll see string entries that look something like this:

Figure 7.2
The String editor with string
table entries



The screenshot shows a window titled "STRINGTABLE : sth_FileNew". It contains a table with three columns: "ID Source", "ID Value", and "String". The table lists 16 entries, each with a unique ID value and a corresponding string description.

ID Source	ID Value	String
sth_FileNew	1	Help on New
sth_FileOpen	2	Help on Open
sth_FileSave	3	Help on Save
sth_FileSaveAs	4	Help on SaveAs
sth_FilePrint	5	Help on Print
sth_FileExit	100	Help on Exit
sth_File	8	Help on File
sth_EditUndo	9	Help on Undo
sth_EditCut	10	Help on Cut
sth_EditCopy	11	Help on Copy
sth_EditPaste	12	Help on Paste
sth_EditClear	13	Help on Clear
sth_EditDelete	14	Help on Delete
sth_Edit	15	Help on Edit
sth_ViewAll	16	Help on View All

Each string table entry requires an ID Source, an ID Value, and the string itself. Before you learn how to enter this information in a string table, you need some background on how Windows handles string resources.

Windows and strings

Each string in a string table must have a unique integer ID. Windows groups strings into segments that contain 16 strings each. Strings with IDs from 0 to 15 make up the first segment, strings 16 to 31 are in the second segment, and so on. When you compile your resources, the strings are added to the executable file in segments. At runtime, these segments are loaded into memory.

When your application requires a particular string, Windows loads the entire string segment that contains that string into memory. Why should you care how Windows handles strings? Because you can reduce the amount of memory your application requires if you plan how you assign string IDs carefully.

Suppose you define 32 strings for your application. If you assign IDs 0 through 31 to these strings, two string segments are added to your executable file. Each time your application needs a string, 16 strings will be loaded into memory, even if several of the strings are not needed.

A better way is to group your strings logically. For example, if one part of your application requires five strings, number them 0 to 4. If a second part of your application requires nine strings, put them into a second segment by numbering the strings 16 to 24. Then Windows can load related strings all at once without loading other strings that won't be needed. Although entire segments will still be loaded, each unused ID only takes up one byte of memory. Therefore, if you place five strings in a segment, the unused IDs will consume only eleven more bytes of memory. If the loaded segment contained eleven unused strings, the amount of memory used would be considerably more.

For more information about using identifiers and identifier files, see page 49.

When you specify a unique string ID, you can use an integer or an alphanumeric identifier (a **#define** in C or C++, or a constant declaration in Pascal) that stands for an integer. If you choose to use alphanumeric identifiers to make the string IDs easier to remember, you probably want to store your identifiers in an identifier file (a header file for C and C++ or a unit or include file for Pascal). Be sure one of these files exists before you try to add identifiers to it from inside the String editor.

Now that you know a bit about how Windows handles string resources and how to specify a unique string ID, you need to learn how to enter a string in a string table.

Entering a new string

Note the headings at the top of the String editor:

- An ID Source contains an integer for the string. If you assign an identifier as the ID, it appears here. Otherwise, you'll see the integer ID.
- An ID Value always contains the integer ID for the string.
- A String is the text string. It's stored in the string table as a Pascal string; that is, a byte indicating the length of the string precedes the text of the string. A string can contain up to 255 text characters.

To enter a new string in a string table,

- If the table is a new one, just start entering information for the string as described in steps 3, 4, and 5 in the steps that follow.
- If you're adding a string to a table, start with step 1 in the steps that follow.

1. Select the string above the line where you want to add the new string.
2. Press *Ins* or choose Stringtable | New Item.
3. Type in an ID Source for the string or accept the number the String editor puts in this field. Based on the ID Source you type, the String editor finds the appropriate integer value for the ID Value.

To restore the ID Source to its original setting, press Esc before pressing Tab or Enter.

You can type an integer or an alphanumeric identifier that stands for an integer. If you type an alphanumeric identifier that doesn't represent an integer, the String editor prompts you to create a new identifier when you press *Enter* or choose a menu command.

To create a new identifier,

- a. Choose Yes. The New Identifier dialog box appears.
- b. Type in a new value for your identifier and specify where you want the identifier stored. Choose OK.

To restore the String field to its original setting, press Esc before pressing Tab or Enter.

4. Press *Tab* or click in the box under String and type the text string.

Each string can be a maximum of 255 characters long and can contain any C-type escape sequences, including the following: \n (newline), \t (tab), \r (carriage return), \ (backslash), \" (double quote).



When the Resource Workshop compiler encounters a C-type escape sequence in a string entry, it produces the corresponding ASCII hexadecimal value in the object code, and it's up to your program to interpret the value correctly. For example, when the compiler parses `\b\040\x7F`, it produces the hex sequence `07207F`. Your code might interpret this sequence as the ASCII characters BEL, SPC, and DEL, or it could assign another meaning to these hex values.

5. Press *Enter* (to accept the new value) or *Ins* (to accept the value and insert a new one).

Editing existing strings

The String editor makes it easy to change individual strings. To select a string with your mouse, click the string you want to edit. Using the keyboard, press *Tab*, *↑*, or *↓* to move through the table. Place your cursor on the string you want to edit.

Changing a string

You can erase the ID Source and String values for any string and then you can type new values. You can't directly change what's displayed in the ID Value field, but the String editor updates it depending on what you type in the ID Source field.

Restoring changed string values

You can use the Undo and Redo features to restore the values of strings you have changed. Choose *Edit | Undo* or press *Alt+BkSp* to undo a change. If you undo too many changes, choose *Edit | Redo* or press *Shift+Alt+BkSp* to redo the changes you've undone.

Deleting a string

To delete a string, select the string and then choose either *Edit | Cut* or *Stringtable | Delete Stringtable Item*.

Editing the resource script of a string table

Besides editing a string in a string table, you can also use a text editor to edit a string in a resource script.

To edit the resource script of a string table,

1. Select the string table in the Project window.
2. Choose *Resource | Edit As Text*.

The resource script text will appear, ready for you to edit.

Changing the string

To edit a string,

1. Find the string you want to edit and make the necessary changes to the string. Change only the text that appears between the quotation marks.
2. Double-click the editor's Control-menu box.
3. In response to the prompt "Resource has changed. Compile?", choose *Yes*.

Resource Workshop compiles your changes for you and saves them. If there's a syntax error, Resource Workshop puts you back in the text editor so you can correct the error.

Changing the identifier Changing the other component of a string entry, the identifier, can be more complicated because the identifier might not exist or might need to be changed to a new integer value. If you change an identifier in the String editor, you're prompted for a new value. If you change it in the text editor, however, you must already have added the identifier to the appropriate identifier file. If you haven't, the compiler attempts to compile the string table and returns the following error: "Expecting unsigned short integer." This error indicates that the compiler tried to interpret the identifier name, but couldn't find an identifier for it.

Of course, you can always insert the actual integer value of the identifier in the ID Source field, but that makes your string table and program less readable and flexible.

If the compiler can't find an identifier because one doesn't exist, you could add a new identifier:

1. Make the Project window the active window.
2. Choose Resource | Identifiers.
3. Choose the New button.
4. In the New Identifier dialog box, enter the new identifier name and its value.
5. If necessary, scroll through the File list until you find the file you want to store the identifier in, select it, and choose OK.
6. Select the Stringtable entry in the Project window and choose the Resource | Edit As Text again.
7. Double-click the Control-menu box again, then choose OK to recompile. Now the compiler finds the identifier and completes compilation successfully.

As you can see, it really is much easier to use the String editor to change or add identifiers to strings.

Saving a string table

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close

the project. Resource Workshop places a STRINGTABLE entry in your resource script in the Project window automatically when you create a new string table resource. To save your string table, you need to save the entire project.

To save the entire project, choose File | Save Project.

Testing a string table

*For information about
compiling resources into an
executable file, see
Chapter 3.*

To test your string table, you'll need to compile the resource script file into an executable file. Then you can run the executable file to see if your strings appear as you think they should.

Creating a sample string table

The example that follows creates a few strings that Resource Workshop uses to describe menu options. It shows you how easy it is to work with string resources using Resource Workshop's String editor.

Without Resource Workshop, you'd use the following resource script to create these strings:

```
STRINGTABLE
BEGIN
    MENU_FILE,    "Create, open, or close files"
    MI_FILENEW,   "Create a new project, resource, or file"
    MI_FILEOPEN,  "Open a resource file"
    MI_FILESAVE,  "Save this resource file"
END
```

The uppercase alphanumeric string preceding each string is a unique identifier for the string. As with all Windows resources, each string requires an integer ID. Without Resource Workshop, you'd have to separately define the associated integer IDs for all these identifiers in a header file (for a C program) or in an include file or unit (for a Pascal program).

Here's how to create these sample strings with the Resource Workshop String editor:

1. Make sure you've already opened a project. If you've been doing the examples using MYPROJ.RC, you can open that project.

For more information on using identifiers and identifier files, see page 49.

2. If you don't already have an identifier file (a header file for C **#defines**, or a unit or include file for Pascal constants) set up for the project, set one up now and call it MYPROJ.H (for this example, we're assuming a C program).
3. Choose Resource | New and tell Resource Workshop to create a new string table. You'll see the String editor.
4. Backspace over the number in the text box under ID Source and type the identifier for the string. For the first string, it's MENU_FILE.
5. Press *Tab* to move to String. Note that you skip over ID Value; you'll return to ID Value in a minute.
6. Under String, type the text of the string. For the first string, it's Create, open, or close files.
7. To define the next string, choose Stringtable | New Stringtable Item.

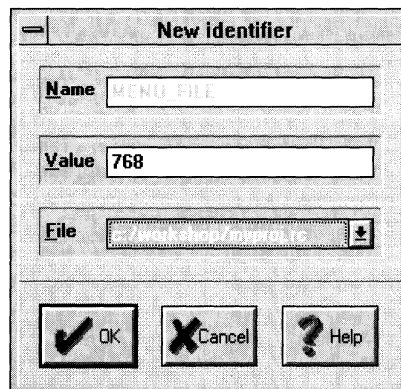
Before the String editor lets you move on to the next string, it checks for an integer ID for the string you've just added. First, it checks what you typed under ID Source. If you entered an integer, the String editor puts the same integer under ID Value.

For more information about identifiers, see page 22.

Since you entered an alphanumeric identifier (MENU_FILE in this case), the String editor checks for a C **#define** or a Pascal constant declaration that points to an integer identifier. Since there isn't one, you're asked if you want to create a new identifier.

8. Choose "Yes" to bring up the New Identifier dialog box.

Figure 7.3
The New Identifier dialog box



9. Enter a unique integer ID in the Value text box. For the first identifier, type 768.
10. Scroll down the File list until you find MYPROJ.H, then select it.
11. Choose OK to accept the new identifier and put it in MYPROJ.H.

Repeat steps 4 through 11 to define the three other strings shown at the start of this section: MI_FILENEW, MI_FILEOPEN, and MI_FILESAVE. Try varying the sequence by pressing *Enter* instead of *Tab* after typing the identifier name.

You'll notice that for each new string, the String editor increments the integer ID by 1 over the last integer ID. You needn't pick this number for the integer ID; the String editor simply puts it there for your convenience.

When you've finished creating the four strings, your string table should look like this:

Figure 7.4
The String editor with four strings defined

STRINGTABLE: MENU_FILE		
ID Source	ID Value	String
MENU_FILE	768	Create, open, or close files
MI_FILENEW	769	Create a new project resource or file
MI_FILEOPEN	770	Open a resource file
MI_FILESAVE	771	Save this resource file

As a last step, you can close the string table by double-clicking on the Control-menu box for the String editor window. The String editor gives the new table the name of the first identifier in the table. If the first ID Source entry had been a number, that number would have become the name of the string table. A string table's name can be changed only by changing the first ID Source entry in the table.

- This naming convention makes sense if you recall that strings are loaded in segments of 16 strings each, and the integer ID of a string indicates where the string occurs in a segment. The integer ID of the first string in the table indicates where the table starts in a segment.

Using the Paint Editor

This chapter is useful for subsequent chapters that describe how to create various kinds of bitmapped images.

The Resource Workshop Paint editor is a tool you can use when you create or edit any bitmapped resource, including the following standard bitmapped resources:

- Icons
- Cursors
- Bitmaps
- Fonts

See the README file on the distribution disks for information on where to find some sample bitmaps, cursors, and icons.

Starting the Paint editor

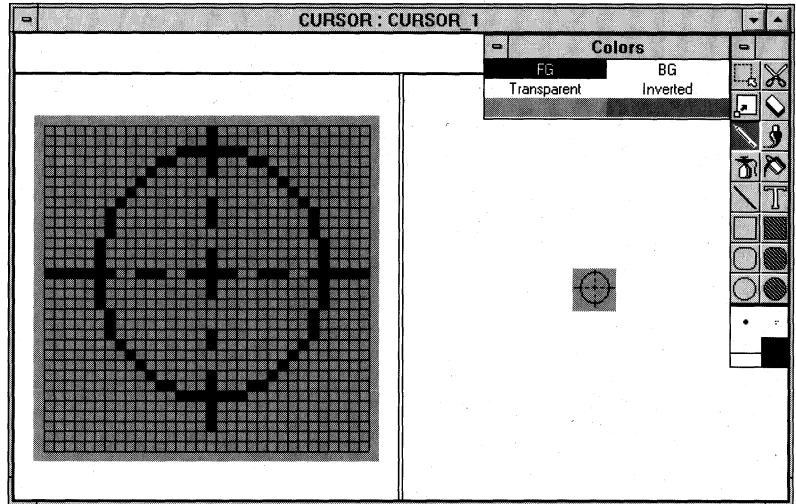
When you create or open a file containing a bitmapped resource, Resource Workshop loads the resource and automatically puts you in the Paint editor. Loading cursors, fonts, and bitmaps works differently from loading icons, as the next two sections describe.

Loading cursors, fonts, and bitmaps

To load a cursor, font, or bitmap, either double-click the resource entry in the Project window or select the resource entry and choose Resource | Edit. Resource Workshop starts the Paint editor and displays the resource, ready for editing.

For example, if you open the sample project RWPDEMO.RC and double-click the cursor resource CURSOR_1 in the project window, Resource Workshop displays this cursor in the Paint editor.

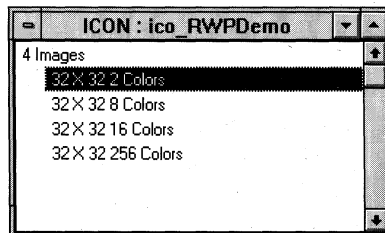
Figure 8.1
The RWPDEMO cursor



Loading icons

When you double-click an icon resource in the Project window (or select it and choose Resource | Edit), you see an Icon window before you see the Paint editor. This window lists different images of your icon. You might, for example, create two versions of an icon: a 2-color version for monochrome monitors and a 16-color version for color monitors. When loading an icon resource, Windows chooses the version that most closely matches the current display driver.

Figure 8.2
Icon Window for the
RWPDEMO icon



See Chapter 9 for more
information on editing icons.

In the Icon window, to select an Icon image and start the Paint editor, you can

- Double-click an existing image of the icon
- Select an image and choose Images | Edit
- Use Images | New Image to create a new image of the icon

For example, if you open the sample project RWPDEMO.RC and choose the icon ico_RWPDEMO from the project window, you see the different images of the icon design listed in the ICON window. If you want to edit the 32 pixel by 32 pixel, 16-color image, you can double-click the 32×32 16 Colors entry in the ICON window to display the Paint editor.

Features available to resources

The Paint editor supplies features commonly used to edit all bitmapped resources as well as features particular to each type of bitmapped resource. The primary way it does this is to isolate to one selection on the menu bar the features that change. You'll notice that the menu bar remains the same for all commands except one: The third command from the right (between "Options" and "Window"). The name of this command corresponds to the resource type being edited, and the menu selections listed when you choose the command change according to the resource type.

For example, if you're creating a cursor, you can specify the cursor's hot spot by choosing Cursor | Set Hot Spot. If you're editing a font, you can specify its size by choosing Font | Font Size.

Most of the features described in this chapter are available to all bitmapped resources. The Paint editor functions specific to the different resource types are described in Chapters 9 through 11.

The basic Paint editor functions generally work the same, no matter what type of bitmapped resource you're working with. The common Paint editor functions described in this chapter include the following:

- The Tools palette, which contains
 - Two types of selection tools, one for rectangular areas and one for irregular areas, to use in selecting a portion of an image so you move, copy, delete, or paste it
 - Paint tools to paint and edit images
 - A Text tool to add text to your image
 - A Zoom tool to enlarge or reduce the view of an image

- A group of style selections to view and change brush shapes, line width, and pattern
- A Colors palette to choose colors for your image
- Two window panes to show two different views of the resource

Understanding foreground and background colors

See page 185 for more information on foreground and background colors.

Before you learn about the tools, it might be helpful to understand what *foreground* and *background* colors are.

- The *foreground* color is the color you draw with the left mouse button and is typically one of the colors you use to create features of your drawing, such as lines, boxes, shading, and so on.
- The *background* color is the color you draw with the right mouse button and is usually the color that appears to underlie your drawing. It's also the color that's left behind when you select an area and delete or move it.

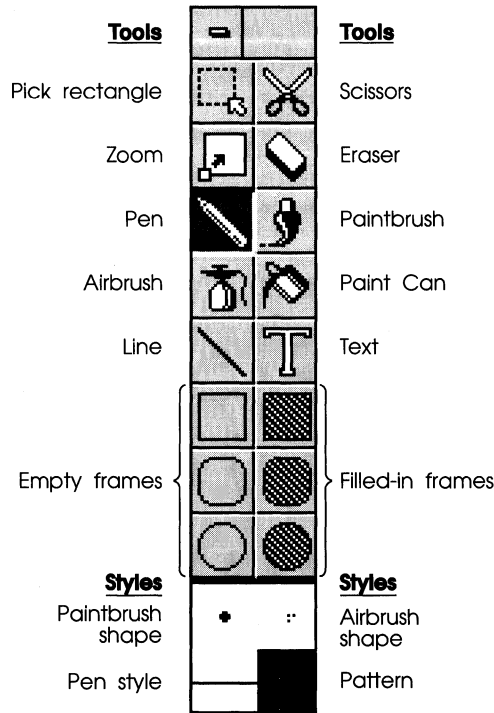


If you're using the eraser tool, the buttons are reversed: the left button produces the background color and the right button produces the foreground color.

Using the Tools palette

When you open a resource in the Paint editor, the Tools palette is in the upper right of the edit window. You use the Tools palette to choose the Paint editor tool you want to work with. At the bottom of the Tools palette are four style selections for brush shapes, fill patterns, and line widths.

Figure 8.3
The Paint Editor Tools palette



The Tools palette is similar to a window: you can move it around the screen, and you can close and open it.

- To close the Tools palette, you can
 - Double-click the Tools palette's Control-menu box or choose Hide from the Tools palette's Control menu.
 - Choose *Resource-type* | Hide Toolbox, where *Resource-type* is the Paint editor menu bar selection corresponding to the type of resource being edited (for example, Icon).
- To open the Tools palette after you've closed it, choose *Resource-type* | Show Toolbox.

To choose a tool, use the left mouse button to click the tool you want. The sections that follow introduce you to each tool.

The Pick Rectangle tool



*Choose Edit | Select All to
select the entire image.*

The Pick Rectangle tool is used to select a rectangular area of your image for copying, moving, or deleting. To select an area, you press the left mouse button and drag the mouse until the flashing outline includes the area you want, then release the mouse button. Clicking either mouse button or pressing *Enter* or *Esc* outside the flashing outline deselects the area.

Once you've selected an area, you can use the commands in the Edit menu to cut, copy, delete, duplicate, or paste into the selected area, or you can use the mouse to move or duplicate the area.

- Both the Cut and Copy commands put the image in the Windows clipboard so you can copy it to another bitmap, either in the same project or in another project running in another copy of Resource Workshop. However, their effect on the current image is different.
 - Cut deletes the current image.
 - Copy leaves the current image intact.
- The Delete command deletes the selected area by filling it with the background color. Pressing *Del* has the same effect.
- The Duplicate command duplicates an area of the image by copying the area you've selected and placing the copy in the upper left corner of the image. As long as the area is still selected, you can use the mouse or arrow keys to move the copied area anywhere in the image.

Instead of using the Duplicate command, you can use the mouse to duplicate a selected area and move it. Select the area, then hold down *Shift* and press the left mouse button and drag the copy to the new location.

- To move the selected area, press the left mouse button anywhere in the area and drag to a new location.
- The Paste command copies the contents of the clipboard into the current window, placing the top left corner of the clipboard image into the top left corner of the current edit window.

If you have an area selected, the Paint editor pastes the clipboard image into the selected area. The Paint editor stretches or shrinks the contents of the clipboard as necessary to try to make it fit the selected area.

The scissors



The scissors perform basically the same function as the Pick Rectangle tool: selecting an area of an image. However, with the scissors you can select and move areas of any shape, not just rectangles.

To select an area, press the left mouse button and drag the scissors until the flashing outline includes the area you want, then release the mouse button. You can cut, copy, delete, duplicate, or move the selected area just as you can with the Pick Rectangle tool (as described in the previous section).

The Zoom tool



See page 180 for more information on zooming.

Zooming the entire image

The Paint editor uses the center of the image as a reference when zooming the entire image.

Use the Hand tool or scrollbars to move the zoomed image around.

You can use the Zoom tool to zoom the entire image in and out, or you can outline an area of an image that you'd like to zoom, and have Resource Workshop zoom in on that area.

To zoom in on the entire image, double-click on the Zoom icon in the Tools palette. Resource Workshop increases magnification to the next higher multiple of 400%, but no more than 1600%. You can also choose View | Zoom In to perform the same function on the currently selected window.

To zoom out on the entire image, hold down the *Shift* key, then double-click the Zoom icon. Resource Workshop decreases magnification to the next lower multiple of 400%, but no less than the actual size of the image (100%). You can also choose View | Zoom Out to perform the same function on the currently selected window.

Zooming a portion of the image

To zoom a portion of the image, select the area you want to magnify by clicking the left mouse button and dragging the Zoom tool. When the flashing outline includes the area you want to see, release the mouse button. Resource Workshop zooms the area to the largest zoom percentage that will fit in the window pane.

For example, if you zoom in on a relatively large area, it might only fit if it's zoomed to 200% or 300%. But if you choose a very small area, it might zoom to as much as 1600%.

To zoom the area back out, either choose View | Zoom Out or hold down the *Shift* key and double-click the Zoom icon in the Tools palette.

The eraser



The eraser works like a square paintbrush (see the description on the next page). When you press the left mouse button and drag the eraser across your image, it reveals the current background color. To erase an entire image, double-click the eraser in the Tools palette.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

- Use the left mouse button to reveal the current background color (BG on the Colors palette).
- Use the right mouse button to reveal the current foreground color (FG on the Colors palette).

Notice that because you're using the eraser to reveal colors, the buttons on the mouse are the opposite of other tools you use to paint. For example, with the paintbrush, you use the left mouse button to paint the foreground color. But with the eraser, the left mouse button reveals the background color.

Before you use the eraser, you might want to check the current colors in the Colors palette. (See "Working with colors" on page 184 for more information.)

The pen



The pen is used to paint free-form lines and shapes, either as quick sketches in an unzoomed image or a pixel at a time in a zoomed image. To sketch with the pen tool, press a mouse button and drag the pen across your image. When you've finished sketching, release the mouse button.

The line you paint can be either the current foreground color or the current background color.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

You can paint straight or irregularly shaped lines. (If you really want to paint straight lines, use the Line tool instead of the pen.)

Before you paint a line, you might want to specify the line style and foreground and background colors. (See “Choosing a line width” on page 195 and “Working with colors” on page 184 for more information.)

The paintbrush



The paintbrush is used to paint free-form patterns that vary more in width and type than those you can produce with the pen. To paint, you press a mouse button and drag the paintbrush across your image. As you drag the paintbrush, Resource Workshop paints a free-form line. When you’ve finished painting, release the mouse button.



When you choose the paintbrush, you get a cursor that represents the current brush shape. The cursor is the same size as the area actually painted only for the actual-size image (the 100%, unzoomed image). If you zoom an image, the area painted is larger than the cursor.

You can paint with either the current foreground color or the current background color.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you use the paintbrush, you might want to specify the brush shape, pattern, and colors. (See “Choosing brush shapes” on page 193, “Choosing paint patterns” on page 194, and “Working with colors” on page 184 for more information.)

The airbrush



Like the paintbrush, the airbrush paints free-form patterns on your image. However, the airbrush works more like spray paint than a paintbrush. If you drag it slowly across your image, it paints a thick pattern, but if you drag it quickly, it paints a scattered, thinner pattern.

When you choose the airbrush, you get a cursor that represents the current brush shape. The cursor is the same size as the area actually painted only for the actual-size image (the 100%, unzoomed image). If you zoom an image, the area painted is larger than the cursor.

To use the airbrush, you can either press a mouse button once and drag the airbrush, or you can click it repeatedly, as if you were repeatedly pressing the nozzle of a spray can.

The airbrush color can be either the current foreground color or the current background color.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you use the airbrush, you might want to specify the airbrush shape, pattern, and colors. (See “Choosing brush shapes” on page 193, “Choosing paint patterns” on page 194, and “Working with colors” on page 184 for more information.)

The paint can



The paint can is used to fill an area of your image with the currently selected color. The paint can fills in any single area of your image that is a single color.

For example, if you use the paint can on an empty image, the paint can fills the entire image. If you use the pen to draw a line that splits the image in half, the paint can fills whichever half you choose.



If you use the paint can on an area that's not entirely surrounded by other colors, the color leaks out into other parts of the image that are the same color as the original area.

To use the paint can, place the paint can's cross hairs in the portion of the image you want to fill, then click a mouse button.

The paint can color can be either the current foreground color or the current background color.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you use the paint can, you might want to specify the current colors. (See “Working with colors” on page 184 for more information.)

The Line tool

For free-form lines, use the pen.



If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

The Line tool is used to paint straight lines. Press the mouse button and drag the Line tool across your image. When you've finished drawing the line, release the mouse button.

If you want the lines you paint to be limited to 45-degree increments, hold down *Shift* as you paint. With *Shift* down, you can only paint a horizontal or vertical line or a line on a 45-degree angle from the horizontal or vertical.

The color you paint with the Line tool can be either the current foreground color or the current background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you paint a line, you might want to specify the line style and current colors. (See "Choosing a line style" on page 195 and "Working with colors" on page 184 for more information.)

The Text tool



To add text to your image, choose the Text tool and click where you want the text to begin. A flashing cursor appears and you can begin typing text.

Before you use the Text tool, you might want to specify how and where you want the text to be displayed, as follows:

- Use Text | Font to specify the typeface, size, and style of the text.
- Use the Text | Align commands to specify how the text is aligned.

For more information about using the Text menu commands, see "Adding text to a resource" on page 190.

You needn't choose any Text menu commands before you enter the text; you can also choose them immediately after you enter the text (before you click the mouse again). For example, you can choose the typeface and size before you type any text. If you notice as you type that the text is too large to fit in your image, you can choose the Font command again to decrease the size of the text.

Text is always displayed in the current foreground color. Before you type text, you might want to specify the foreground color by clicking the left mouse button on the color you want in the Colors palette. Just as with the typeface and size, you can change the current text color if you do so immediately after you enter text. (See “Working with colors” on page 184 for more information on selecting colors.)

Painting empty frames



If you press Shift while drawing with one of these tools, you get a square or a circle.

If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

There are three tools you can use to paint empty frames in your image: the rectangle, the rounded rectangle, and the ellipse.

These tools paint a frame using the current line style and foreground color.

To paint an empty frame, first choose the tool with the appropriate shape. Next, point the tool's cross hair where you want to start a corner of the frame, press a mouse button and drag the frame tool until the frame outline surrounds the area you want, then release the mouse button.

The color of your empty frame can be either the current foreground color or the current background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you paint a frame, you might want to specify the frame width and the color. (See “Choosing a line style” on page 195 and “Working with colors” on page 184 for more information.)

Painting filled-in frames



If you see FB in the Colors palette, the same color is selected as the current foreground and background color.

There are three tools you can use to paint filled-in frames in your image: the filled rectangle, the filled rounded rectangle, and the filled ellipse.

These tools paint a frame using the current line style. The frame is filled using the current pattern and color. Specify a null pen width if you don't want Resource Workshop to paint a frame around the filled-in pattern.

To paint a filled frame, first choose the tool with the appropriate shape. Next, point the tool's cross hair where you want to start a corner of the frame, press a mouse button and drag the frame tool until the frame outline surrounds the area you want, then release the mouse button.

The color of both the frame and the pattern inside the frame can be the current foreground color or the current background color.

- Use the left mouse button to paint using the current foreground color (FG on the Colors palette).
- Use the right mouse button to paint using the current background color (BG on the Colors palette).

Before you paint a filled frame, you might want to specify the line style, color, and pattern. (See "Choosing a line style" on page 195, "Working with colors" on page 184, and "Choosing paint patterns" on page 194 for more information.)

The Hand tool



You can also use the scroll bars to move the image around.

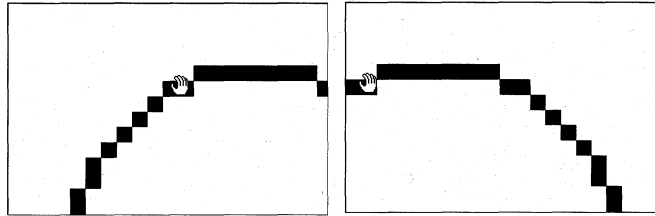
Sometimes when you display a zoomed image, not all of it fits in the display. You can use the Hand tool to move the image around so you can see other parts of it. Unlike other tools, the Hand tool isn't included in the Tools palette. But you can temporarily change any tool (except the Text tool) into a hand by holding down **Ctrl**.

You can think of the hand as a *grabbing* tool because you just click the hand on the image and drag it in the direction you want it to move.

For example, suppose the top left corner of an image is displayed and you want to see the top right corner. You can hold down **Ctrl** to turn the current tool into a hand, then click to grab the right

side of the displayed image and drag it to the left. As you drag to the left, the left side of the image moves out of view, and the right side of the image moves into view. When you release *Ctrl*, the current tool returns.

Figure 8.4
Grabbing a zoomed image

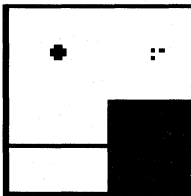


Grab the image...

...and move it to the left.

For more information about working with zoomed images, see “Zooming” on page 180.

The style selections



At the bottom of the Tools palette is a box containing the four styles that indicate the paintbrush shape, the airbrush shape, the line styles, and the patterns you can paint.

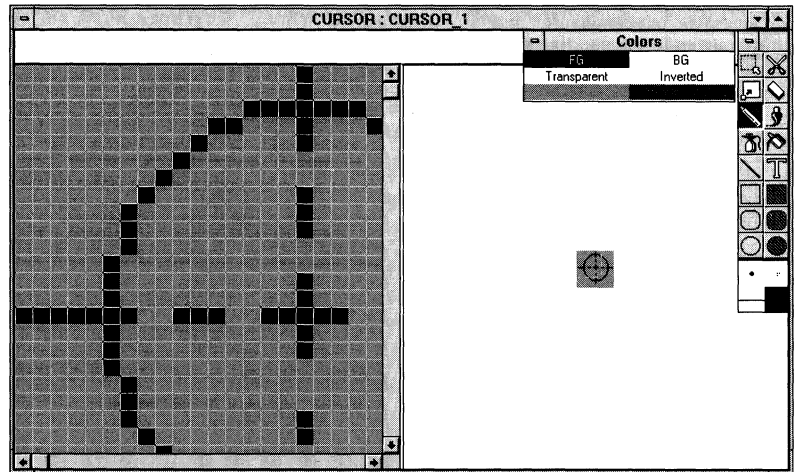
You can click any style you want to change, or you can use menu commands to choose styles. (For more information about choosing styles, see “Choosing brush shapes” on page 193, “Choosing paint patterns” on page 194, and “Choosing a line style” on page 195.)

Using the two window panes

In the Paint editor, you can look at two different views of the image you’re creating or editing. You can split the window vertically to show the two views side-by-side, or you can split the window horizontally to show one view above the other. You can also choose how to zoom each view.

For example, you could split the window vertically and display the entire image at its actual size in the right window pane, and zoom in on a small portion of the image in the left window pane.

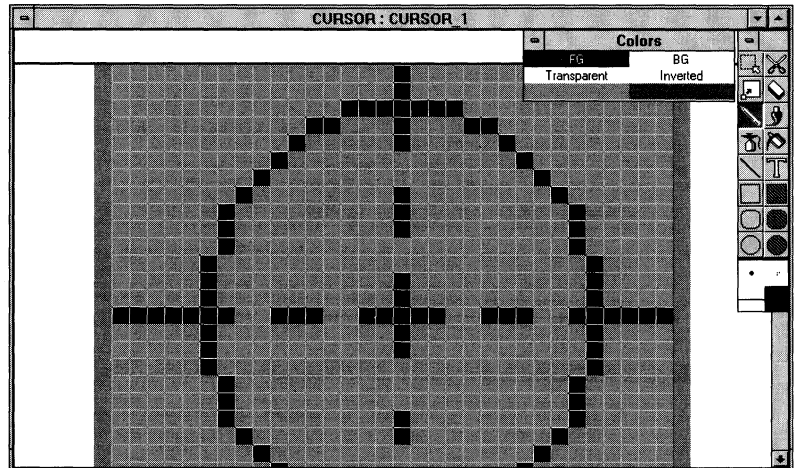
Figure 8.5
The Paint Editor's split screen



To select one of the images, click the mouse in that image.

If you want to see more of one view than the other, you can move the cursor to the line that splits the images (the separator bar) and, when the cursor becomes a double arrow, drag the separator bar. For example, with the windows split vertically, you might want to drag the separator bar to the right to see more of a zoomed image. If you drag the separator bar all the way to the right, you have only a single image.

Figure 8.6
Zoomed image taking entire
edit area



In the previous figure, the separator bar was dragged all the way to the right, and the pane that was on the right is now completely

gone. You can use the View menu to get it back and to control the way the panes are displayed in the Paint editor.

Figure 8.7
The View menu

Zoom in	Ctrl+Z
Zoom out	Ctrl+O
Actual size	Ctrl+A
C<u>G</u>A resolution [32X16]	
Spl<u>i</u>t h<u>o</u>r<u>i</u>z<u>o</u>n<u>t</u>a<u>l</u>	
Spl<u>i</u>t v<u>e</u>r<u>t</u>i<u>c</u>a<u>l</u>	

Use Split Horizontal and Split Vertical to control how the panes are positioned or to see two views of the resource again. The other commands in the View menu let you choose how to zoom images, as described in the next section.

For icons and cursors, there's an additional command in this window, CGA Resolution [32×16]. See page 205 for an explanation of how to zoom icons, and page 226 for an explanation of how to zoom cursors.

Zooming

Resource Workshop can show you an image at its actual size or it can zoom an image up to 1600%. You can use the Zoom tool or the View menu to zoom either of the two window panes in the Paint editor.

See page 171 for more on the Zoom tool.

With the Zoom tool, you outline an area of an image that you'd like to zoom, and Resource Workshop uses an appropriate zoom percentage to show you that area. If you use the Zoom tool, Resource Workshop can zoom an image in multiples of 100% up to 1600%.

Double-clicking (zoom in) and Shift+double-clicking (zoom out) on the Zoom icon has the same effect as using the menu.

You can also zoom in and out using the View menu. With the View menu commands, you can zoom an image to 400%, 800%, 1200%, 1600%, or back to its actual size.

Since there are two panes, how do you determine which one zooms in or out when you choose one of the View menu commands? It's simple: Resource Workshop keeps track of the current window pane (the one you worked with most recently). If you drew most recently on the left image, that's the image that will get bigger when you choose View | Zoom In. You can also use any tool to click anywhere in a window pane to make it the

current one. If a zoomed image takes up the whole window pane, you might want to use a tool that won't paint something accidentally, such as the Pick Rectangle tool or scissors.

Using the zoom accelerators

The accelerators listed in the View menu make zooming much quicker. For example, if you want to zoom in or out repeatedly until you've adjusted the image to your liking, it's much easier to press *Ctrl+Z* or *Ctrl+O* a few times than to keep choosing menu commands. You can also double-click the Zoom icon in the Tools palette to zoom in and *Shift*+double-click it to zoom out. Pressing *Ctrl+A* to returns the image to its actual size.

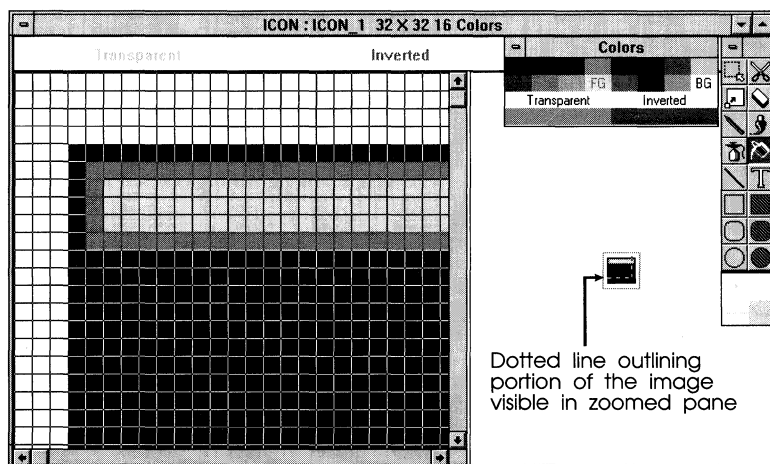
Table 8.1
Zoom commands

View command	Accelerator key	Mouse action on Zoom icon
Zoom in	Ctrl+Z	Shift+double-click
Zoom out	Ctrl+O	Double-click
Actual size	Ctrl+A	None

Seeing which part of the image is zoomed

If you zoom an image to a size that's too large to fit in the pane and display the image at its true size in the other window pane, Resource Workshop places a dotted rectangle over the unzoomed image. This dotted rectangle indicates the portion of the image currently displayed in the zoomed window pane.

Figure 8.8
Zoomed image and dotted outline on unzoomed image



Moving a zoomed image around

See page 177 for more information on the Hand tool.

If a zoomed image is too large to fit in a window pane, Resource Workshop displays scroll bars that let you move the image around so you can see different parts of it.

You can also use the Hand tool to grab an image and move it. To do this, press *Ctrl* so the current tool becomes a Hand tool, then click the image and drag it in the direction you want it to move. You might have to repeat this process several times if you want to move a zoomed image more than the total width or length of the window pane.

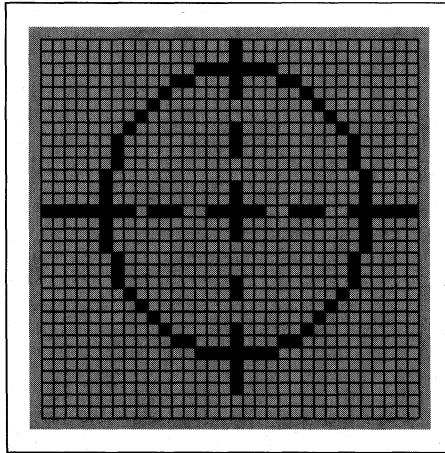
See Figure 8.7.

If the image is also displayed unzoomed in another window pane, you can watch the dotted outline move as you look at different parts of the zoomed image.

Displaying a grid in a zoomed window

To help you control images on a pixel-by-pixel basis, you can display a grid on any zoomed image by choosing Options | Editor Options and clicking the Grid On Zoomed Windows option. Each square of the grid highlights a single pixel.

Figure 8.9
A zoomed image with a grid overlay



Reading the status line

At the bottom of the Paint editor is a divided status line that displays information about commands and paint tools.

The right side: current paint tool information

The right side of the status line tells you which paint tool you're using and where it is on the screen. You might also see color information, depending on the tool you're using.

The tool status message you see depends on which tool you've selected and where the tool is on your screen. Here are two examples of messages with accompanying explanations.

Line x: 18 y: 32

This message indicates that you've selected the Line tool, which is located at the pixel coordinates 18,32. Pixel coordinates are counted from the upper left corner of the image. As you move towards the right side of the image, the X value increases; as you move towards the bottom of the image, the Y value increases.

Brush x: 20 y: 37 R: 128 G: 0 B: 0 Palette Index: 1

This message indicates that you've selected the paintbrush, which is located at the pixel coordinates 20,37. The R, G, and B values indicate the red, green, and blue color values of the color at those coordinates. The color of the image at 20,37 is currently the color 1 in the Colors palette index, which has a value of 128 red.

You can look at palette index numbers and RGB values by double-clicking a color in the Colors palette.

See "Customizing colors" on page 188 for more information on RGB values and palette indexes.

You can also see the palette index and RGB settings for the color at the cursor if you select the pen, the airbrush, or the paint can.

The left side: menu command explanations

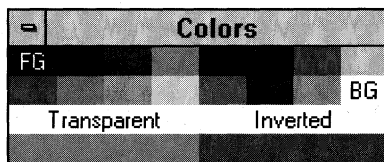
As you click a menu or use accelerator keys to choose a menu command, the left side of the Paint editor status line displays more information about the currently highlighted command.

For example, choose the View menu and drag the cursor with the Zoom In command. The status line shows *Magnify active window*. When you highlight another of the commands in the View menu (either by dragging with the mouse or using the arrow keys), the status line displays an explanation of that command. For example, if you highlight the Zoom Out command, the status line displays *Reduce active window*.

Working with colors

As you edit a resource in the Paint editor, you can use the Colors palette to choose the colors you want. (This book isn't printed in color, but you can see them on your screen.)

Figure 8.10
The 16-color Colors palette



You can work with a Colors palette even if your image is black and white, and you can hide or display the Colors palette any time.

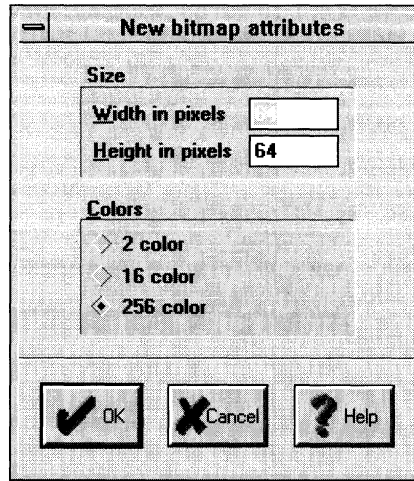
You can use the Colors palette to

- Choose a foreground color
- Choose a background color
- For icon and cursor resources only, choose transparent and inverted areas

Choosing the number of colors for a resource

When you create a new bitmap or icon, Resource Workshop displays a dialog box that lets you choose how many colors you want to include in your resource. For example, when you create a new bitmap, Resource Workshop displays the New Bitmap Attributes dialog box.

Figure 8.11
The New Bitmap Attributes
dialog box



While you're editing a bitmap or icon, you can change the number of colors in the image using the **Size and Attributes** command. This command is located in the **Bitmap** menu or the **Icon** menu, depending on the type of resource you're currently editing.

For bitmap and icon resources, you can include up to 256 colors in your resource. The number of colors you can use (and see in the **Colors** palette) depends on the type of display driver you're using with Windows.

For example, the standard Windows VGA driver supports a fixed set of only 16 colors. If you want to see colors other than those in this fixed set, you have to use a different display driver, such as the **Video 7** driver that comes with Windows (this driver also requires a **Video 7** VRAM card or equivalent with at least 512K of memory) or, if you have a super-VGA board, the **Extended VGA (EVGA)** driver supplied by the manufacturer of the board.

Using a foreground color

To use a foreground color,

1. Click the left mouse button on the color you want in the **Colors** palette. You see the letters *FG* appear on that color (unless it's already selected as a background color, in which case you see *FB*).

2. From the Tools palette, choose a tool that draws or paints, and with the left mouse button click the tool or drag it to draw or paint with the foreground color.

The eraser operates in the opposite fashion from the drawing tools. Dragging it with the left mouse button produces the background color, and dragging it with the right mouse button produces the foreground color.

Using a background color

To use a background color,

1. Click the right mouse button on the color you want in the Colors palette. You see the letters *BG* appear on that color (unless it's already selected as a foreground color, in which case you see *FB*).
2. From the Tools palette, choose a tool that draws or paints, and with the right mouse button click the tool or drag it to draw or paint with the background color.

The eraser operates in the opposite fashion from the drawing tools. Dragging it with the left mouse button produces the background color, and dragging it with the right mouse button produces the foreground color.

Including transparent and inverted areas in a cursor or icon

You can pick inverted and transparent as background and foreground "colors."

Transparent and *inverted* areas show up as "see through" parts of your icon or cursor at runtime.

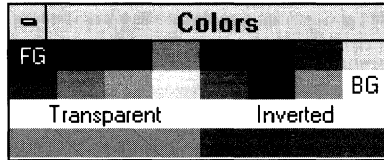
- A transparent area allows the color of the area behind the icon or cursor to show through.
- An inverted area behaves like a transparent area except that it inverts the color over which you place the icon or cursor. For example, when you put the inverted part of an icon on a black area, the black color behind the inverted area displays as white. If you move the inverted part of the icon to a white area, the white color behind the inverted area displays as black.

Using transparent and inverted attributes

You pick transparent and inverted attributes the same way you pick any color in the Colors palette (they appear as the choices *Transparent* and *Inverted*), and you draw and paint transparent and inverted areas the same way you draw and paint with other foreground and background colors.

Unlike foreground and background colors, however, you don't have complete control over what will be displayed in transparent and inverted areas when your program uses the icon or cursor; the color displayed depends on the color of the area behind the icon or cursor.

Figure 8.12
Colors palette with
Transparent and Inverted
colors



The long rectangles that represent Transparent and Inverted also show what color will display for transparent and inverted areas when you edit your icon or cursor. If the rectangle under Transparent is cyan, all transparent areas in the image you're drawing will be cyan, and all inverted areas will be the inverse of cyan, or red.

For more information about adding transparent and inverted areas to your cursor and icon resources, see Chapter 9, "Creating icons," and Chapter 10, "Creating cursors."

Showing and hiding the Colors palette

If you want to hide the Colors palette, you can close it by double-clicking the system menu icon in the upper left corner of the palette.

You can also use the Hide Palette and Show Palette commands. The menu where you'll find these commands depends on the type of resource you're currently editing. For example, if you're editing an icon, you'll find the Show Palette and Hide Palette commands in the Icon menu.

Only one of these commands appears at a time. If the Colors palette is visible, you'll see only the Hide Palette command. The Show Palette command appears only if the Colors palette is hidden.

Customizing colors

If you're editing a color bitmap or icon, you can modify the Colors palette to include any colors supported by your display driver. It won't make sense to do this if the Colors palette already includes all the colors supported by your computer. But if your display driver is capable of displaying 256 colors and you're working with a 16-color image, you can include any of the 256 colors in the 16-Colors palette.

The following sections describe how to edit any of the colors in the Colors palette, including transparent and inverted.

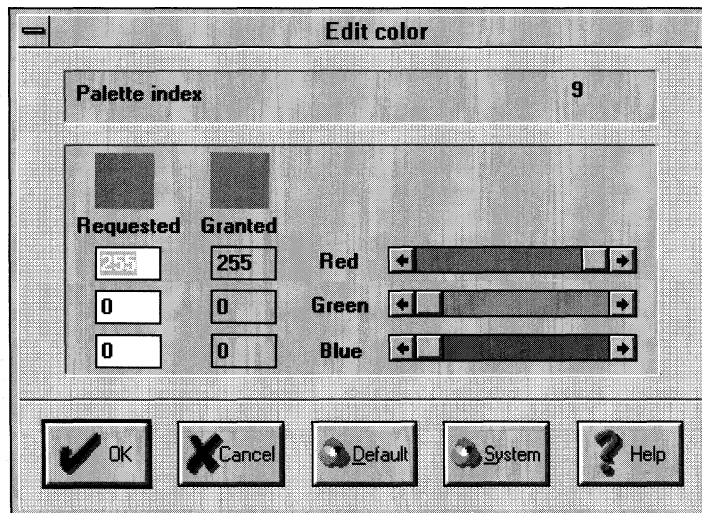
Editing colors in the Colors palette

To edit a color, do one of the following:

- Double-click it.
- Select it as the foreground or the background color, then choose either Icon | Edit Foreground Color or Icon | Edit Background Color.

Resource Workshop brings up the Edit Color dialog box.

Figure 8.13
The Edit Color dialog box



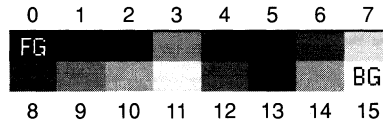
The palette index To help you identify where the color goes in the Colors palette, the Edit Color dialog box shows you the palette index at the top.

The Colors palette boxes are numbered from left to right across each row, starting with zero at the top left box. The transparent and inverted boxes aren't included.

For example, in the default 16-color palette, the colors are numbered as shown in the following figure:

Figure 8.14

The 16-color palette index



The RGB values indicate the color's red, green, and blue values.

You can also see the palette index and RGB settings for a color at the cursor in the edit window if you select the pen, the paintbrush, the airbrush, or the paint can, and then move the cursor over the image and look at the right side of the status line.

Editing a color

To edit a color, you can change its RGB values either by typing new values in the left column or using the slide bars on the right side of the dialog box. Resource Workshop displays the closest matching color for the new RGB values in the color box on the left (for a 16-Colors palette, you might see a dithered color appear in this box). In the color box on the right, Resource Workshop matches the RGB values to the closest color permitted by the limits of the current palette. You only see this right box change color if the closest match is different from the current color.

If you enter a value in the left column, click on another box in the column to get the new value to take effect.

For example, if you're using the standard Windows VGA driver and you edit the red color, you'll see that its RGB values are 255, 0, 0. If you change green to 50 and blue to 125, you see the left box change colors while the color in the right box stays the same. (Its RGB values remain at 255, 0, 0 because those values represent the color closest to 255, 50, 125.) If you then change blue to 200, you'll see the color in both boxes change, the left box to a dithered color and the right one to magenta (RGB values 255, 0, 255), the Windows VGA color closest to 255, 50, 200.

When you're finished changing colors, click OK or press *Enter* to put the new color in the Colors palette.



Before you leave the Paint editor you must turn off the Save with Default Device Colors option to allow Resource Workshop to save your customized palette. Choose Options | Editor Options and make sure there isn't a check mark next to Save With Default Device Colors.

Resource Workshop can save customized colors only for a 256-color palette.

Default button

The Default button retrieves the color from the default palette (a Windows stock object) that has the same index as the Palette index (shown on the top of the dialog).

System button

The System button retrieves the color from the system palette that has the same index as the Palette index (shown on the top of the dialog). This button is disabled for VGA displays, since the VGA device driver does not support logical palettes.

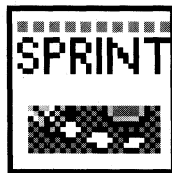
Changing colors of transparent and inverted areas

Changing the colors used to display transparent or inverted areas is similar to changing a regular color, except you only get transparent and inverted “colors” with icons and cursors, and the dialog box that comes up changes two colors at a time. Because the methods for bringing up this dialog box differ depending on whether you’re editing a cursor or an icon, this subject is covered in the chapters on those resources (Chapter 9, “Creating icons,” and Chapter 10, “Creating cursors”).

Adding text to a resource

You can add text to any resource that was created with the Paint editor. For example, you might want to add text to an icon to represent the program name.

Figure 8.15
An icon that includes text



To add text to a resource, click the Text tool (the capital *T*) and then click in the image where you want the text to start. Resource Workshop displays a flashing text cursor. Then type the text you want.

To determine how the text is displayed, you can use the Text menu commands either before or immediately after you type the

text. For example, after you type your text, you can choose Text | Align Center to center the text on the insertion point and Text | Font to display the Select Font dialog box and change the font and size of the text.

In addition, immediately after you type the text (and before you click the mouse again), you can change its color by selecting a new color from the Colors palette. You can also press the right mouse button if you want the text to be displayed using the current background color.

➡ If you click another tool or click in another area of the image after entering text, you can't change anything in the text you've just entered. At that point, the text becomes just another part of the bitmap as though you had painted it there.

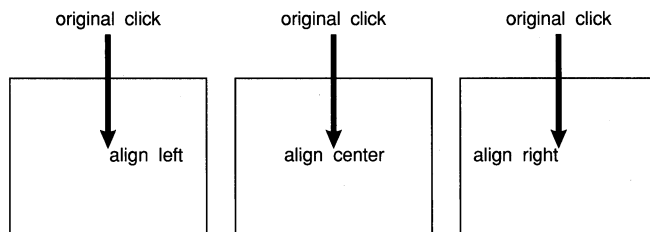
Aligning text

To align text, use the Text | Align commands.

Each of the Align commands controls where the text is displayed in relation to where you initially clicked before typing the text. You can use an Align command either before you type text or immediately after you finish typing.

For example, suppose you choose Text | Align Left and then use the Text tool to type five characters on your image. As you type, the characters move towards the right side of your image because they are aligned where you originally clicked on the left. If you choose Text | Align Center, the letters center on the location of the original click. If you choose Text | Align Right, the characters move to the left of your original click position.

Figure 8.16
Aligning text



➡ You can't change the alignment of text you've typed once you click to make another selection.

Choosing fonts, size, and text style

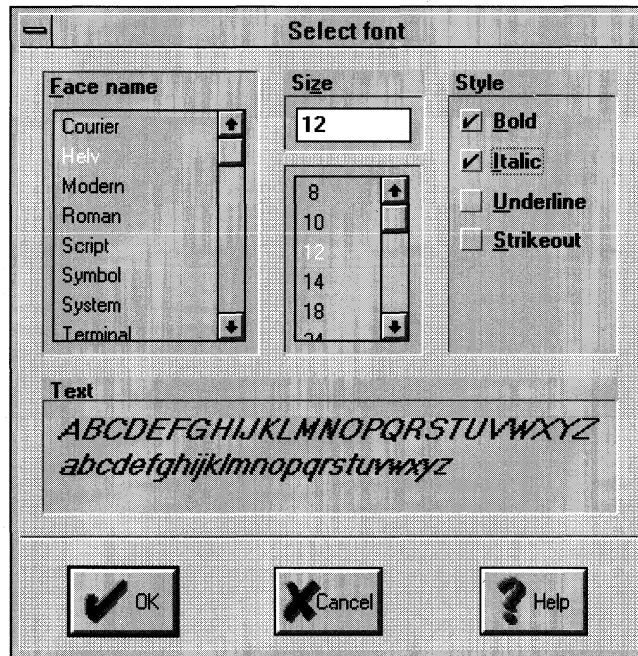


To choose how text is displayed, use the Text | Font command either before you type text or immediately after you finish typing.

You can't change the font of text you've typed once you click to make another selection.

After you choose the Font command, Resource Workshop displays the Select Font dialog box.

Figure 8.17
The Select Font dialog box



You can choose the typeface, size, and style of text that you want. For example, you might want Helvetica as the typeface, 10 point as the size, and bold underlined as the style.

Notice the characters displayed at the bottom of the Select Font dialog box. They change to show you the current typeface and size you've selected, but they do not show the styles you've selected.

Choosing brush shapes

You can paint images in a resource using the paintbrush or the airbrush. Resource Workshop lets you choose the shape of the brush or airbrush.

The current paintbrush and airbrush shapes are always displayed as the top two of the four styles at the bottom of the Tools palette.

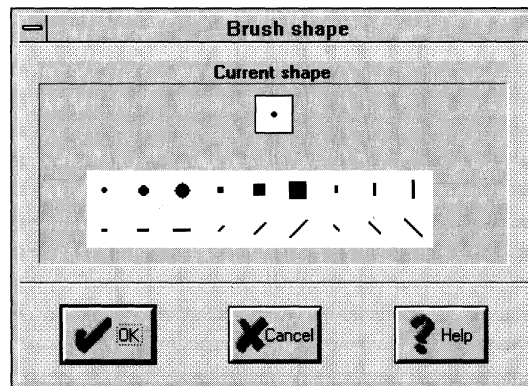
You can choose a brush shape or an airbrush shape in one of the following ways:

- For a paintbrush shape, use Options | Brush Shape.
- For an airbrush shape, use Options | Airbrush Shape.
- Click the paintbrush or airbrush shape style selection.

You'll see a dialog box that shows you all the possible brush shapes you can choose. At the top of the dialog box, Resource Workshop shows you the shape that's currently selected.

For example, if you're changing the paintbrush shape, here's the dialog box you'll see.

Figure 8.18
The Brush Shape dialog box



The next time you use the paintbrush or airbrush, Resource Workshop uses the shape you specify. This shape stays the same until you specify a new one.

Choosing paint patterns

You can choose a pattern to paint on your image. The tools that let you do this are the following:

- The paintbrush
- The airbrush
- The filled rectangle
- The filled rounded rectangle
- The filled ellipse

For example, if you choose a pattern that consists of widely spaced vertical lines, that pattern appears if you paint an area with one of the previously listed tools. However, if you use a tool not affected by your pattern choice, such as the paint can, the painted area will be a solid color.

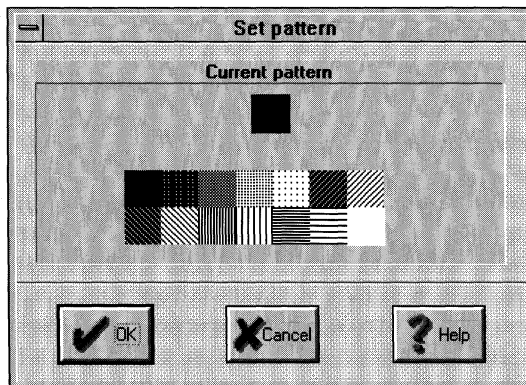
The current pattern is always displayed in the lower right corner of the Tools palette.

You can choose a pattern in one of the following ways:

- Choose Options | Pattern.
- Click the style selection for the pattern.

You'll see a dialog box that shows you all the possible patterns you can choose. At the top of the dialog box, Resource Workshop shows you the pattern that's currently selected.

Figure 8.19
The Set Pattern dialog box



The next time you use a tool capable of filling with a pattern, Resource Workshop uses the pattern you specify. This pattern stays the same until you specify a new one.

Choosing a line style

You can control the line style produced by any of the following tools:

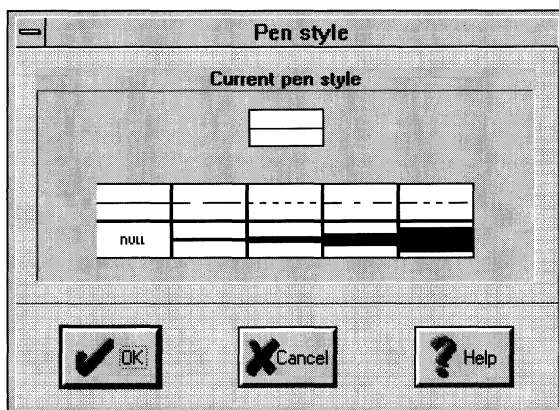
- The pen
- The Line tool
- The unfilled rectangle
- The unfilled rounded rectangle
- The unfilled ellipse

You can choose a line style in one of the following ways:

- Choose Options | Pen Style.
- Click the line style selection in the Tools palette.

You'll see a dialog box that shows you all the possible styles you can choose. At the top of the dialog box, Resource Workshop shows you the style that's currently selected.

Figure 8.20
The Set Pen Style dialog box



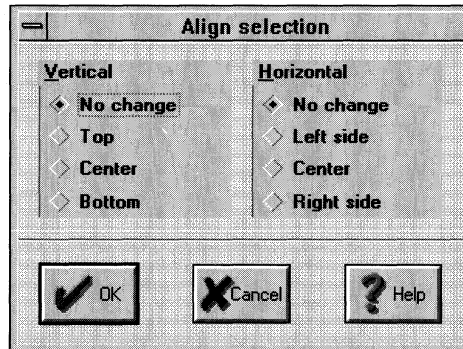
Notice the null choice for a pen width. You can use null when you want to paint filled-in frames that don't have a border.

The next time you use a tool capable of painting a line, Resource Workshop uses the width you specified. The pen width you choose stays the same until you specify a new one.

Aligning a selected area

You can align a selected area of an image with the top, bottom, sides, or center of the current edit window. Aligning the selected area moves it to the location you specify, just as if you had selected the area and moved it with the mouse. To align an area you've selected, choose Options | Align and choose the options you want from the Align Selection dialog box.

Figure 8.21
The Align Selection dialog
box



Choose the options under Vertical to align the selected area along the vertical axis of the edit window and the options under Horizontal to align it along the horizontal axis.

For example, if you choose Top and Left Side and click OK, the selected area moves to the top left corner of the window. If you choose Vertical | Center and Right Side and click OK, the selected area is centered equidistantly between the top and bottom of the window and is moved as far to the right as possible.

The Paint editor uses the background color to paint the area where the selected area was located, just as it would have done had you moved the selected area with the mouse.

Resizing a selected area

You can resize or move an area of an image you've selected with the Pick Rectangle tool by choosing Options | Size. If you resize the area, the Paint editor stretches or compresses the image inside the selection area accordingly.

When you choose Options | Size, the Paint editor displays the Stretch Selection dialog box.

Figure 8.22
The Stretch Selection dialog box

Old position/size		New position/size	
Left	0	Left	
Top	0	Top	0
Width	52	Width	52
Height	33	Height	33

This dialog box shows the pixel coordinates of the top left corner of the currently selected rectangular area (the Left and Top selections) as well as its current width and height in pixels. To move or resize the area or both, pick pixel values that fall within the current window and enter them in the boxes under New Position | Size. Press *Tab* to move from field to field.

For example, to move a rectangular area to the top left corner and make it 30 pixels wide and 5 pixels high, do the following:

1. Select an area of the image using the Pick Rectangle tool.
2. Enter 0 as the Left value and press *Tab*.
3. Enter 0 as the Top value and press *Tab*.
4. Enter 30 as the width and press *Tab*.
5. Enter 5 as the height and press *Tab*.
6. Click OK or press *Enter*.

You see the currently selected box positioned in the top left corner with the new width and height. In addition, the Paint editor paints the original selection rectangle with the background color, just as it would have done had you moved the selection rectangle with the mouse.

Setting global Paint editor options

To set the global Paint editor options, choose Options | Editor Options to display the Set Paint Editor Options dialog box.

Figure 8.23
The Set Paint Editor Options
dialog box



Draw on both images When you have two views of the same image displayed in the Paint editor, you can choose to have Resource Workshop update each image as you draw.

If you don't choose this option, Resource Workshop updates the other image only after you finish drawing an element. For example, as you drag a brush across the image, Resource Workshop shows the line only on the image on which you're actually drawing. However, once you release the mouse button, Resource Workshop then updates the other image, too.

Grid on zoomed images On zoomed images, you can display a grid that shows you how the image is painted on a pixel-by-pixel basis. See "Zooming" on page 180 for more information on this option.

Save with default device colors Uncheck this option if you want to save a custom Colors palette you've created. When this option is checked, any colors you've customized in the Colors palette will revert to the default color when you close the Paint editor.

You can save customized Colors palettes only if your display driver is capable of displaying 256 colors and supports logical palettes. See page 188 for information about customizing the Colors palette.

Creating icons

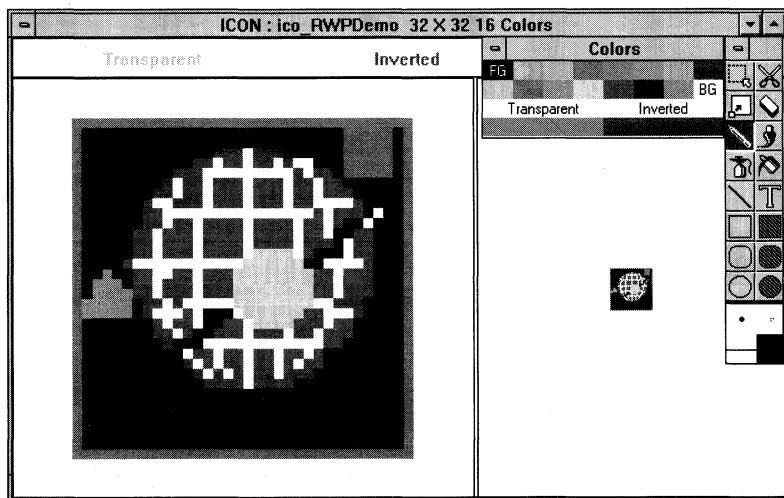
Icons are small bitmapped images, generally 32×32 or 32×16 pixels in size. Windows programs typically use customized icons to represent minimized windows. The following icon comes up when you minimize the sample program RWPDEMO:

Figure 9.1
The RWPDEMO icon



To design your icons, you use the Resource Workshop Paint editor.

Figure 9.2
The Paint Editor with the
RWPDEMO icon loaded



The Paint editor includes various paint tools and an easy-to-use Colors palette for selecting colors. It also lets you zoom your image and shows you multiple views of the icon you're creating. Chapter 8 explains how to use the Paint editor.

When working with icons, you perform four primary tasks:

- Starting the Paint editor
- Customizing an icon
- Testing the icon
- Saving the icon

The first task, starting the Paint editor, displays a workspace that can contain an icon ready for you to customize. The second and third tasks, customizing the icon and testing it, are functions of the Paint editor itself. The fourth task, saving the icon, happens automatically if you save your project file. If you want to save an icon in a separate file, you have a few more steps to follow.

Additional tasks discussed in this chapter are how to add an image to an icon resource, how to delete an icon or image, and how to edit the resource script of an icon.

A sample project on page 213 illustrates how to create and customize an icon.

Starting the Paint editor

To work on an icon, you need to start the Paint editor by creating a new icon or choosing one that already exists.

There is also a third option: creating a new image for an icon that already exists. Because this option is typically used just to add a new color format, it's covered later in "Adding an image to an icon resource" on page 211.

See page 213 for an example of a resource script.

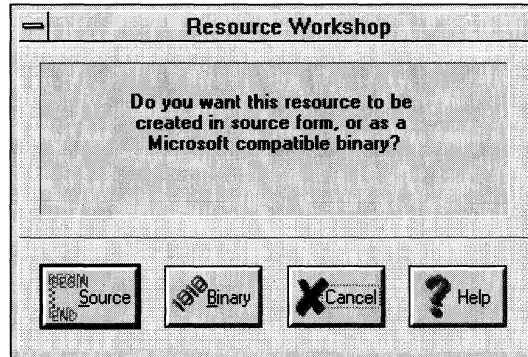
If you want to edit the resource script of an icon, select the icon from the project window by clicking it, then choose Resource | Edit As Text. It's unlikely you'll want to edit an icon as a resource script because the script is almost entirely a series of hexadecimal values.

Creating a new icon

To create a new icon,

1. Make sure you've already opened a project. See Chapter 3 if you need help with opening a project.
2. Choose Resource | New. Resource Workshop displays the New Resource dialog box. In the Resource Type window, scroll down to ICON and select it.
3. The current project file is highlighted in the box under "Place resource in." You can scroll down this list to pick another file (if any is listed) or click OK to accept the current project file.
4. Resource Workshop next asks whether you want to create the icon in source form (as a resource script) or in Microsoft-compatible binary format.

Figure 9.3
Dialog box prompting for
source or binary



The section that follows gives more information about selecting an appropriate format for your icon.

5. After you've selected your storage format, to start the Paint editor, you can double-click on the new image in the Icon window or select the new image and choose Images | Edit Image.

Selecting a storage format

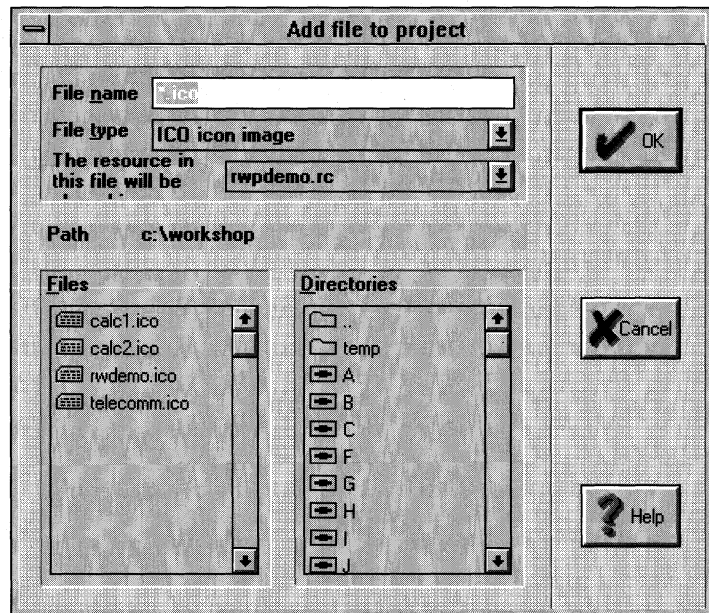
Saving your icon as a resource script puts it directly in your project file. If you want to store the icon in a separate file that's referenced in your project file (for example, because you'd like to share the icon across several projects) or you need your icons to be compatible with the Microsoft Resource Compiler, store your

icon in binary format. Depending on which you choose, Resource Workshop displays a different dialog box.

Binary format

If you choose Binary, Resource Workshop displays the Add File To Project dialog box.

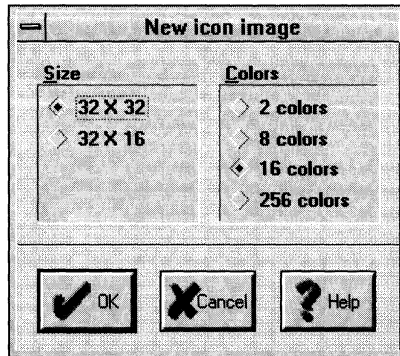
Figure 9.4
The Add File To Project
dialog box



This dialog box prompts you for a file in which to store the icon. Under Files, it displays all the files in the current directory that have the extension .ICO. You can use the Directory box to choose a new directory.

Either enter a new file name or path, or choose a file from the list. You need not use the .ICO extension (although if you don't, it might be hard for Resource Workshop to find the icon file). Resource Workshop supplies you with the correct file type, *ICO Icon image*. When you've made your selections, click OK. Resource Workshop displays the New Icon Image dialog box. (If you've picked an existing file, you'll first be asked if you really want to overwrite it.)

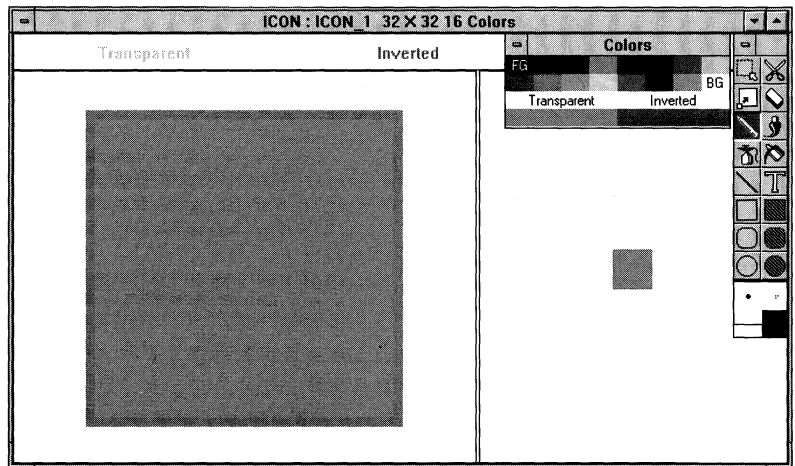
Figure 9.5
The New Icon Image dialog
box



See page 184 for a
description of color formats.

Choose the image size and color format you want, then click OK. Resource Workshop puts the new icon name in the Project window and displays the Paint editor window so you can begin painting the new icon.

Figure 9.6
Paint editor window for a
new 16-color icon



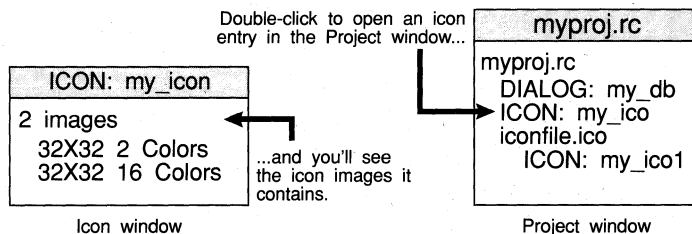
Source format

If you choose Source, Resource Workshop displays the New Icon Image dialog box (see Figure 9.5). Choose the image size and color format you want, then click OK. Resource Workshop puts a new ICON entry in the Project window and displays the Paint editor window (see Figure 9.6).

Editing an existing icon

To edit an existing icon,

1. Open either a new project or an already existing one. (Chapter 3 explains everything you need to know about opening projects.) Resource Workshop then displays the project window.
2. Each icon resource can contain multiple images (typically the same icon in different color formats). To edit one of the images in an Icon resource, open the resource entry (double-click it, or choose Resource | Edit). If there is more than one image in the resource, Resource Workshop displays the Icon window, which lists all the images.



3. In the Icon window, find the icon image you want to edit. Double-click it, or select it and choose Images | Edit Image. Resource Workshop displays the icon in the Paint editor.
4. Once you've displayed the icon in the Paint editor, use the paint tools to customize the icon.

Customizing an icon

After you have the icon open in the Paint editor, you can begin customizing it by using all the Paint editor features described in Chapter 8. Before you start, however, there are a few design issues and other matters for you to consider.

Design issues

It's a good idea to consider the following questions before starting work on your icon:

■ What should the icon represent to the user?

An icon should clearly convey what it's intended to represent. In many cases, icons make the user think of the basic features of the application. For example, you could design an icon to look like a document for a word-processing program or a paintbrush for a paint program. Your icon could also be your company's logo.

■ Is your design simple enough to fit into a 32×32 or 32×16 pixel area?

Remember, your icon is going to be small, so you won't be able to pack a lot of detail into it.

■ Will the icon look good no matter what background the user chooses to display it on?

Think about where the icon is likely to be displayed and test your icon by displaying it on various background colors and patterns.

■ While you're painting the icon, will it be possible to distinguish between colors you use in the icon and the colors assigned to transparent and inverse areas?

Try to pick colors for the transparent and inverse areas of your icon that don't appear as actual colors in the icon itself.

Once you've decided on the appropriate strategy, you can begin customizing your icon.

Zooming the icon

See page 180 for more on zooming.

You zoom icons just like any other bitmapped resource. However, there is an additional command on the View menu, CGA Resolution [32 ×16], that lets you see how the icon would look on a CGA screen. Choosing this command puts the image in actual size and displays it in 32×16 format. To get out of CGA view mode, choose any of the zoom commands.



All you change with this command is the view of the icon, not the icon itself: if it was a 32×32 icon, it still is.

Working with transparent and inverted areas

As mentioned in Chapter 8, transparent and inverted areas show up as colors in the icon when you're editing it. When you use the icon, however, the transparent areas show up as the current desktop color and the inverted areas as the inverse of that color.

The background color

A new icon or cursor image automatically has a transparent background. So, if you create a new image and draw only a single line in it, you see only that line when you test the image against the desktop color; you don't see a box with a background color and a line on it.



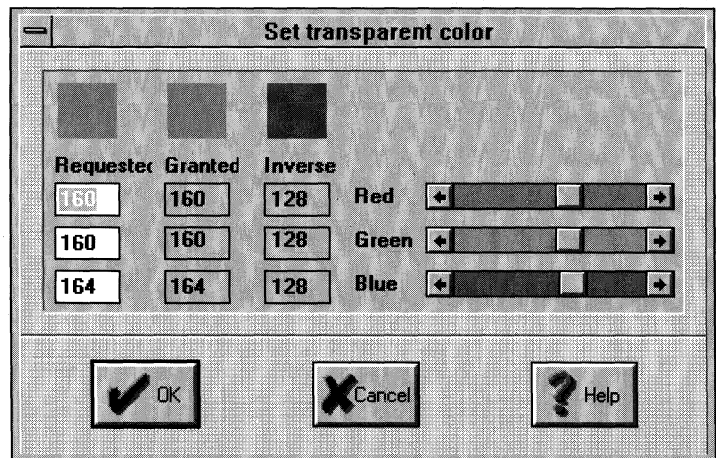
If you want a background color instead of a transparent background for the image, your first steps in drawing it should be to select a background color, choose the paint can, and click with the right mouse button in the image to fill it with the background color.

Changing the color of transparent and inverted areas

You can change the colors the Paint editor displays for transparent and inverted areas as follows:

1. Select Transparent or Inverted by doing one of the following:
 - In the Colors palette, double-click the bar under either Transparent or Inverted.
 - Select either Transparent or Inverted as the foreground or the background color. Then choose either Icon | Edit Foreground Color or Icon | Edit Background Color.
2. Resource Workshop brings up the following dialog box:

Figure 9.7
The Set Transparent Color dialog box



This dialog box works similarly to the Edit Colors dialog box (see Figure 8.13 on page 188), except that it changes both the transparent and the inverted color at the same time.

You'll notice in this box that there is a third color (Inverted) that's the inverse of the color in the middle box. The color in the middle box becomes the color for transparent areas, and the color in the box on the right becomes the color for inverted areas.

Making an icon look three- dimensional

Three-dimensional objects in Windows have an imaginary light source coming from the upper left. For this reason, icons typically have *drop shading* on the right and bottom (more on drop shading later), and three-dimensional buttons have white borders on the left and top and dark borders on the right and bottom.

If you want to create the illusion of this light source and obtain a three-dimensional look, you might want to consider the following first:

- Unless you have some artistic talent, it's best to stick with simpler, flat images, which still can be quite dramatic (for example, take a look at the various Resource Workshop icons).
- Three-dimensional icons usually lose much more than simpler, bolder, two-dimensional icons in the translation from color VGA to monochrome, EGA, and plasma laptop displays.
- A bold, black outline isn't always the best choice for a three-dimensional illusion; try different shades to see which looks best.

If you do decide to create a three-dimensional image, one method to use is *drop shading*. For example, to make a black box look three dimensional, you can draw a gray border on the right side and below the box, as shown here.

Figure 9.8
A black box icon with gray
drop shading



Testing an icon

The easiest way to test an icon is to minimize the Paint editor window in which you're editing the icon, either by selecting the Minimize button in the top right corner of the window or choosing Icon | Test. The icon for this minimized window is the icon you're working on.



If you can't see the icon after minimizing the window, it might be because you have other windows open. Try choosing Window | Tile to make room for the icon at the bottom of the Resource Workshop desktop.

You can also bind the icon resource to an executable file and then run the file to see what the icon looks like. For information about binding resources to executable files, see Chapter 3.

Two primary reasons for testing an icon are

- To see how a color icon looks in black and white when you move it around (Windows turns it into a cursor when you move it).
- To see how transparent and inverted areas look against various backgrounds.

The first test is easy to perform: just minimize the icon editor and move the icon around.

The second test takes a little more work, because Windows carries the current application workspace color around with the icon, making it impossible to test the color icon against various colors without changing the application workspace color. You can change the colors of the transparent and inverted areas to get some idea of what the color combinations look like. However, to fully test the icon, you need to minimize the icon editor and then change the color of the application workspace.

To test the icon against a series of different application workspace colors,

1. If necessary, reduce the size of Resource Workshop so it doesn't take up the entire screen, then move it to the bottom of the screen.
2. Minimize the Paint editor that contains the icon. Make sure the icon is visible and is in a spot where the Resource Workshop application workspace is its background.

3. Change tasks to the Program Manager.
4. Choose the Control Panel. When it appears, move it to the top of the screen.
5. Choose Color.
6. From the Colors dialog box choose Colors palette.
7. Click in the area immediately surrounding the box with "Window Text" in it. You should see "Application Workspace" appear in the Screen Element box above the Colors palette.
8. Click on a color, then click OK.
9. Change tasks back to Resource Workshop and check the icon against the new background color.

At this point, you should be able to see both Resource Workshop and the Control Panel dialog box at the same time. You can select another color by simply clicking on Colors in the Control Panel dialog box and performing steps 5 through 8. Whenever you change the color of the application workspace, Windows changes the color in the Resource Workshop window, where you can check the icon against the new color.

Saving an icon

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. The primary method for saving changes you've made to an icon resource is to save the entire project. You can also save the icon resource separately as a file, but you use this option only under special circumstances.

Saving the project

This is the Save option you'll use most often.

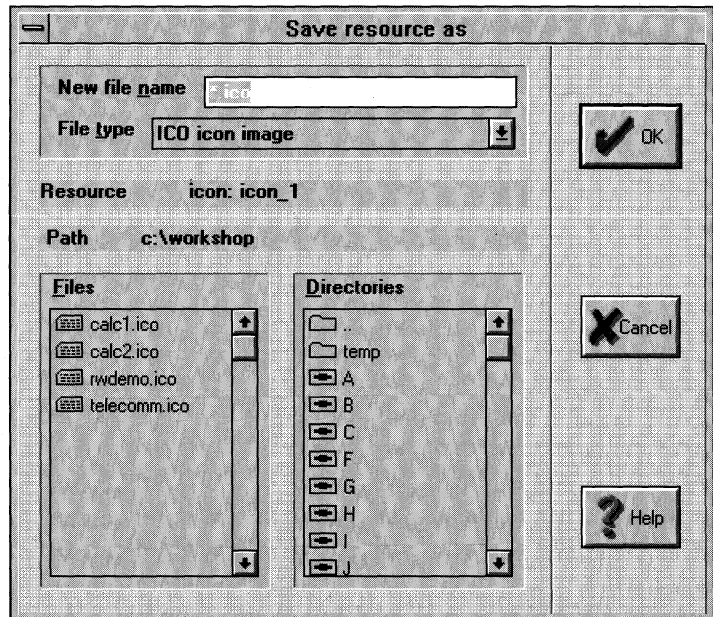
To save the entire project, choose File | Save Project. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. Any changed resource linked to an external file is updated in its external file.

Saving the icon resource as a file

Saving the entire project also saves the icon resource you're working on. However, you might want to save the icon resource as a file because you want to put it in binary format for use by other programs. In this case, you choose Resource | Save Resource | Save Resource As.

1. When you choose Resource | Save Resource As, Resource Workshop displays the Save Resource As dialog box.

Figure 9.9
The Save File As dialog box
for an icon resource



2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or enter the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or click OK.
3. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the icon to be saved in binary format to the external icon file and not in the project file or in any previous icon file.

Adding an image to an icon resource

In general, you create a new icon resource for each icon design (called simply an *icon*), and you don't put different icons in the same icon resource. However, it's likely you'll want to put different color formats of the same icon in one icon resource. These color variations on the same icon are called *images*. For example, if you want a 2-color and a 16-color version of the same design, you can store both versions in the same icon resource.

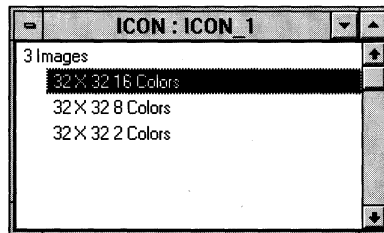
The reason the icon resource supports different color formats is that Windows picks a color format based on the ability of the display hardware to support the format. Windows picks a 2-color format for a monochrome display driver and a 16-color format for the standard Windows VGA driver.

➡ Windows 3.0 doesn't fully support the 256-color version of an icon, even if your display hardware supports it. Your program must supply its own support for 256 colors.

Here's how to add a new image to an existing icon resource.

1. Make sure you've already opened a project (see Chapter 3 if you need information on how to do this).
2. Double-click the ICON entry in the project window, or select the ICON entry and choose Resource | Edit. Resource Workshop displays the Icon window.

Figure 9.10
The Icon window



3. To create a new version of an existing icon, choose Images | New Image. Resource Workshop displays the New Icon Image dialog box (see Figure 9.5 on page 203), from which you can choose the image size and color format.
4. Choose the same size as the existing image and a new color format, then click OK. Resource Workshop displays the new image entry in the Icon window.

See page 218 for a description of how to copy a sample icon.

5. Double-click the new icon, or select it and choose **Images | Edit Image**. You'll see the Paint editor.
6. Typically, what you do next is open one of the existing icon images and copy it into the new (still blank) image. You might also have to edit the image if the colors are translated in a way that changes the form of the icon.

Deleting icons and images

You can delete an icon resource or an icon image. Deleting an entire icon resource deletes all the images in that resource. Deleting an icon image deletes only that single image in the icon resource.

Deleting an icon resource

To delete an icon resource, select it in the Project window, then do one of the following:

- Press the **Del** key or choose **Edit | Delete** to completely delete it.
- Choose **Edit | Cut** to cut the resource into the Windows Clipboard so you can paste it elsewhere.

Deleting an icon image

To remove an image from an icon resource,

1. Open the icon resource from the Project window (double-click it or select it and choose **Resource | Edit**) to display the Icon window.
2. Select the image entry you want to delete, then do one of the following:
 - Press the **Del** key or choose **Edit | Delete** to completely delete it.
 - Choose **Edit | Cut** to cut the image into the Windows Clipboard so you can paste it elsewhere.

Creating the new icon

*See Chapter 3 for
information about opening a
project.*

To create the new icon, you must first open a project, and then go through a series of steps to bring up the Paint editor with the new, blank image in it.

1. Choose File | New Project to create a new project or File | Open Project to open an existing project. If you've been doing the examples in the previous chapters, open MYPROJ.RC.
2. Choose Resource | New and tell Resource Workshop to create a new resource. When Resource Workshop asks what type of resource you want, choose ICON.
3. Choose Source to store the icon as a resource script in the .RC file.
4. Resource Workshop opens the New Icon Image dialog box. Check 32×32 and 16 Colors, then click OK.
5. Resource Workshop opens the Paint editor.

Changing the transparent color

Before you start drawing, it's a good idea to change the color of the transparent and inverted areas of the icon, because the default transparent color, cyan, is a color you'll be using in the drawing. To change these colors,

1. In the Colors palette, double-click the Transparent color.
2. In the Set Transparent Color dialog box, move the Red slide bar all the way to the right and the Green slide bar all the way to the left.
You see the transparent color change to magenta and the inverse color change to green.
3. Click OK to apply the new colors to the Colors palette.

Drawing the calculator

To draw the calculator icon,

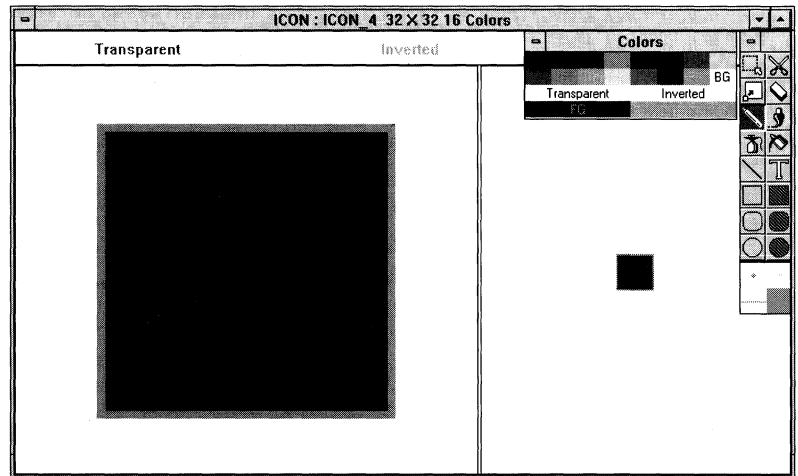
1. Choose Options | Editor Options and check Grid On Zoomed Windows to help you line up the calculator buttons.

The grid shows you the individual pixels when you zoom your icon in the Paint editor. It's easier to see what you're doing if you draw on a zoomed-in view of the icon.

2. Leave the icon displayed at its actual size in the right window. To zoom the image in the left window, double-click the Zoom icon in the Tools Palette until the image is zoomed as large as you can get it with all of it still visible onscreen.

The two views of the icon should look like this:

Figure 9.12
Beginning screen for a new
icon



Each square of the grid on the zoomed view represents a single pixel.

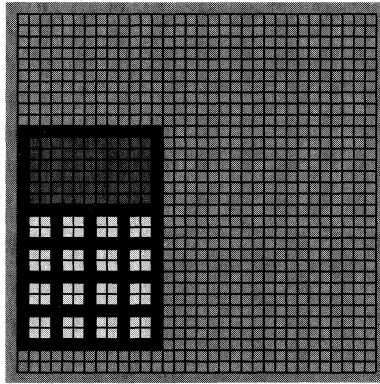
Now you're ready to start drawing.

3. Choose the color dark red for the calculator. When you click in the Colors palette, you see the letters *FG* (foreground) on the color dark red.
4. Click the filled rectangle in the Tools Palette. It's the square tool on the right under the Text tool (the large "T").
5. Draw the calculator in the bottom left part of the icon.

*Be sure to leave two pixels
below the calculator to put
in the shading.*

You can use the filled rectangle to draw all parts of the calculator—the face, the display area, and the buttons. Choose different colors for the calculator display and buttons by clicking in the Colors palette before drawing (try yellow for the keys and dark cyan for the screen). When you've finished, your calculator should look something like this:

Figure 9.13
Zoomed image of calculator
before adding drop shading



*If you erase too much, you
can fix it by choosing Edit |
Undo or pressing Alt+BkSp.*

If you make a mistake, you can use the Undo feature (Edit | Undo or *Alt+BkSp*) to rectify it. You can also use the Pick Rectangle tool or the Scissors tool to outline an area and delete it (use the *Del* key or Edit | Delete) or the eraser to erase it.

If you want to erase everything and start over again, just double-click the Eraser tool in the Tools Palette.

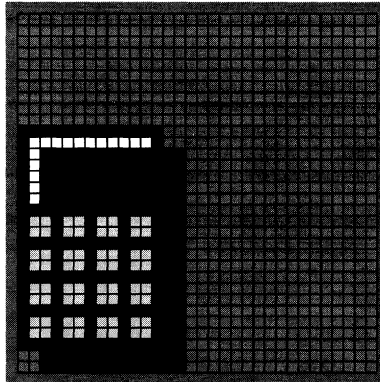
Adding a three- dimensional effect

To add a three-dimensional effect to the calculator,

1. Choose the color black for the shadow.
2. Use the Line tool to fill in a shaded line down the right side and across the bottom of the calculator. The line should be two pixels wide and should start two pixels below the top and end two pixels short of the left side.
3. You can also make the calculator window look three-dimensional by drawing a white line along the window's left edge and across its top.

Here's what the calculator should look like now:

Figure 9.14
Zoomed image of calculator
with shading



When you finish drawing the calculator, choose **File | Save Project** to save your new icon image. It's always a good idea to save early and often.

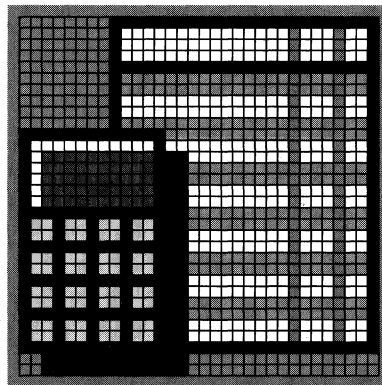
Drawing the ledger page

To draw the ledger page,

1. Choose black, then choose the Line tool.
2. Use the Line tool to draw a vertical black line starting on the fifth pixel to the left of the top, right corner of the calculator. Carry the line up to the top of the edit window, across to its upper right corner, down to within two pixels of the bottom of the Paint Editor, and across to meet the black shading to the right of the calculator. You've just drawn the outline of the ledger page.
3. Choose the color white and the paint can. Click the paint can on the ledger page to fill it with white.
4. Choose light cyan (the lightest blue color) and the Line tool.
5. Click the Line style in the bottom right corner of the Tools Palette and, in the Current Pen Style dialog box, pick the two-pixel line width (the one to the right of Null).
6. Starting two pixels up from the bottom of the ledger page, draw a series of two-pixel-high lines across the width of the visible part of the ledger page. Each line should be two pixels from the previous line. (The bottom three lines will be bounded on the left by the calculator shading.)
7. Choose light red and the Line tool.

8. Click on the Line style in the bottom right corner of the Tools Palette and, in the Current Pen Style dialog box, pick the single-pixel line width (the one directly above Null).
9. Starting three pixels from the right edge of the ledger, draw a vertical line the length of the ledger page. Move four pixels to the left and draw a second red vertical line.
10. Choose black and draw a horizontal line that starts immediately above the topmost cyan area and goes the width of the ledger page.
11. When you've finished your icon, the Paint editor window should look like the following:

Figure 9.15
The Home Budget icon



Copying the image to a different color format

You might want to have different color formats of the same image for different display adapters, such as a black-and-white image for a monochrome adapter. If you want to create a black-and-white version of the icon, instead of drawing it from scratch, you can copy the color version and paste it to a black-and-white version, as follows:

1. First, you need to create a 2-color image. Get to the Icon window (see Figure 9.10 on page 211), then Choose Images | New Image. Resource Workshop displays the New Icon Image dialog box (see Figure 9.5 on page 203).
2. Check the box next to 2 Colors, then click OK. In the Icon window, Resource Workshop displays the new line "32x32 2 Colors" to represent the new image.

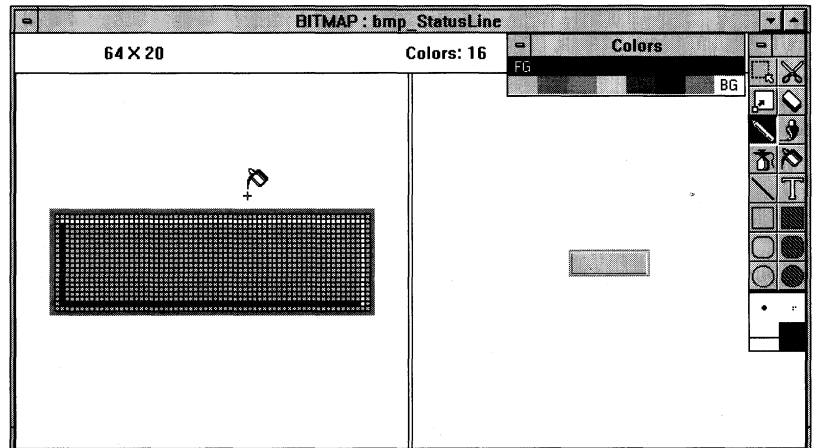
3. In the Icon window, double-click the image entry for the 16-color image you've already created ("32×32 16 Colors").
4. Choose Edit | Select All to select the entire image.
5. Choose Edit | Copy (or press *Ctrl+Ins*) to copy the icon image to the Clipboard.
6. Double-click the Paint editor menu to close it (or press *Ctrl+F4*).
7. In the Icon window, double-click the 2-color icon. Resource Workshop displays the 2-color icon in the Paint editor.
8. Choose Edit | Paste (or press *Shift+Ins*) to paste in the 16-color image of the Home Budget icon. You'll see it appear in black and white.
9. Because Resource Workshop translated the icon from color to black and white, it might have lost some of the detail it had in the color version. If necessary, you can edit the 2-color image to add more detail.

Creating cursors

Cursors are bitmapped images 32×32 pixels in size that represent the current location on the screen. A Windows application often has a number of different cursors that represent different program functions.

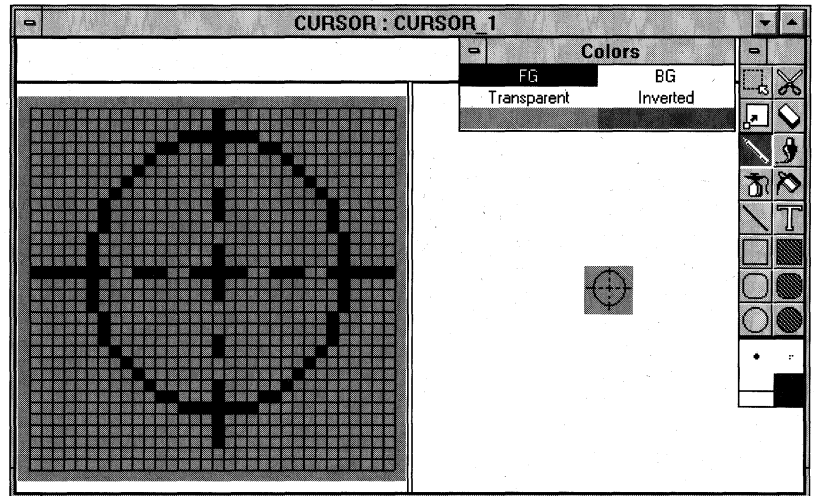
Windows provides a set of standard cursors you can use in your programs. In addition, you can create your own customized cursors to represent different functions of the program. An example of a customized cursor is the Resource Workshop Paint Can cursor that displays when you choose the Paint Can tool in the Paint editor.

Figure 10.1
Paint Editor with Paint Can
cursor



To design your cursors, you use the Resource Workshop Paint editor.

Figure 10.2
The Paint Editor, with a simple
cursor



The Paint editor includes numerous paint tools and a Colors palette that makes it easy to select colors. It also lets you zoom your image and shows you multiple views of the cursor you're creating. Chapter 8 explains how to use the Paint editor.

When working with cursors, you perform four primary tasks:

- Starting the Paint editor
- Customizing a cursor
- Testing the cursor
- Saving the cursor

The first task, starting the Paint editor, displays a workspace that can contain a cursor ready for you to customize. The second and third tasks, customizing the cursor and testing it, are functions of the Paint editor itself. The fourth task, saving the cursor, happens automatically if you save your project file. If you want to save a cursor in a separate file, you have a few more steps to follow.

Starting the Paint editor

To work on a cursor, you need to start the Paint editor by creating a new cursor or choosing one that already exists.

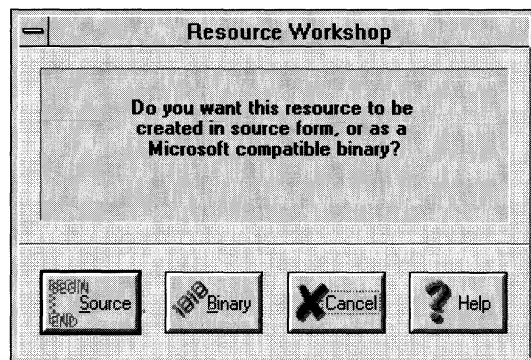
Creating a new

CURSOR

To create a new cursor,

1. Make sure you've already opened a project. See Chapter 3 if you need help with opening a project.
2. Choose Resource | New. In the New dialog box, Resource should be already checked. Click OK.
3. Resource Workshop displays the New Resource dialog box. In the Resource Type window, scroll down to CURSOR and select it.
4. The current project file is highlighted in the box under "Place resource in." You can scroll down this list to pick another file (if any is listed) or click OK to accept the current project file.
5. Resource Workshop next asks whether you want to create the cursor in source form (as a resource script) or in Microsoft-compatible binary format.

Figure 10.3
Dialog box prompting for
source or binary

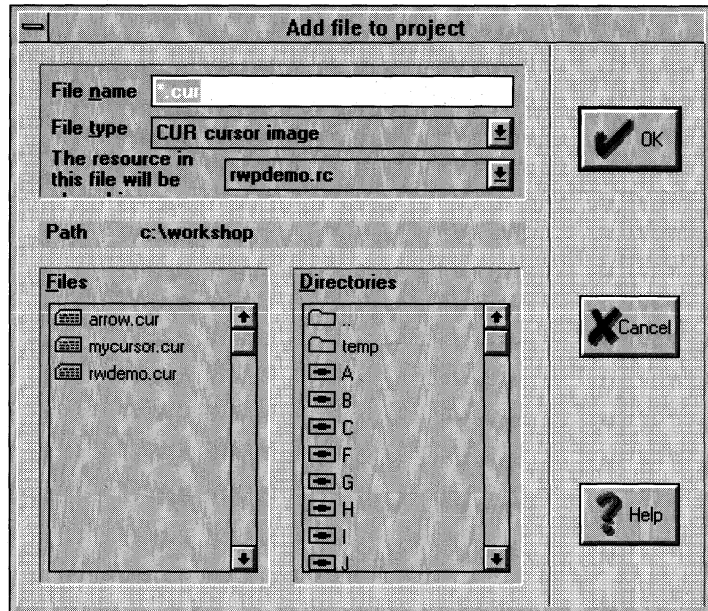


Saving your cursor as a resource script puts it directly in your project file. If you want to store the cursor in a separate file that's referenced in your project file (for example, because you'd like to share the cursor across several projects) or you need your cursors to be compatible with Microsoft compilers (such as the RC compiler), store your cursor in binary format. Depending on which you choose, Resource Workshop displays a different dialog box.

Source format If you choose Source, Resource Workshop puts a new cursor entry with a default title in the Project window and opens the Paint editor with the new cursor resource in it.

Binary format If you choose Binary, Resource Workshop displays the Add File To Project dialog box.

Figure 10.4
The Add File to Project dialog box



This dialog box prompts you for a file in which to store the cursor. Under Files, it displays all the files in the current directory that have the extension .CUR. You can use the Directory box to choose a new directory.

Either enter a new file name or path, or choose a file from the list. You need not use the .CUR extension (although if you don't, it might be hard for Resource Workshop to find the cursor file). Resource Workshop supplies you with the correct file type, *CUR Cursor image*. When you've made your selections, click OK. Resource Workshop displays the new cursor entry in the Project window and opens the Paint editor with the new cursor resource in it.

Editing an existing

CURSOR

Before you start working with a cursor, you need to open either a new project or an already existing one. (Chapter 3 explains everything you need to know about opening projects.) Resource Workshop then displays the Project window.

To display the cursor in the Paint editor, either double-click the cursor entry in the Project window or select the cursor entry and choose Resource | Edit.

See page 231 for an example of a resource script.

If you want to edit the resource script of a cursor, select the cursor from the Project window by clicking it, then choose Resource | Edit As Text. It's unlikely that you'll want to edit a cursor as a resource script because the script is almost entirely a series of hexadecimal values.

Customizing a cursor

After you have the cursor open in the Paint editor, you can begin customizing it. You can use all the features described in Chapter 8. Before you start, however, there are a few design issues and other matters for you to consider.

Design issues

It's a good idea to consider the following questions before you start working on your cursor.

- What should the cursor represent to the user?

A custom cursor should clearly convey what it's intended to represent. A typical use of a custom cursor is to represent the task the user is performing. For example, in a paint program the cursor could be displayed as a brush, an eraser, or a pen, depending on what the user is doing.

- Will it be obvious where the cursor's hot spot is?

The *hot spot* is the cursor's active area, where the user clicks controls, menus, and so on. For example, if you use an arrow for a cursor, the user expects the hot spot to be at the tip of the arrow. With other designs, you could include something like a cross hair to make it clear where the active area is. For example, Resource Workshop's Paint editor turns the mouse cursor into a

paint can with a cross hair when you choose the Paint Can tool (see Figure 10 on page 221).

- Is your design simple enough to fit into a 32×32 or 32×16 pixel area?

Remember, your cursor will be small, so you won't be able to pack a lot of detail into it.

- Will the cursor look good no matter what background the user chooses to display it on?

Think about where the cursor is likely to be displayed and test it by displaying it on various colors and patterns.

When you've decided on the appropriate strategy, you can begin designing a cursor.

Zooming the cursor

See page 180 for more on zooming.

You zoom cursors just like any other bitmapped resource. As with icons, there is an additional command on the View menu, CGA Resolution [32×16], that lets you see how the cursor would look on a CGA screen. Choosing this command puts the image in actual size and displays it in 32×16 format. To get out of CGA view mode, choose any of the zoom commands.



Resource Workshop creates only 32×32 pixel cursors. You can see how the cursor would look on a CGA screen by choosing View | CGA Resolution [32×16], but all you change with this command is the view of the cursor: it's still a 32×32 cursor.

Working with transparent and inverted areas

As mentioned in Chapter 8, transparent and inverted areas show up as colors in the cursor when you're editing it. When you use the cursor, however, the transparent areas allow the colors behind them to show through, and the inverted areas invert the colors behind them.

The background color

A new cursor image automatically has a transparent background. So, if you create a new image and draw a cross hair in it, you see only that cross hair when you test the image; you don't see a box with a background color and a cross hair on it.

Just as with an icon, you can change the apparent color of the transparent and inverted parts of the cursor. Typically, the

background of a cursor remains transparent so the user can see where the cursor is located.

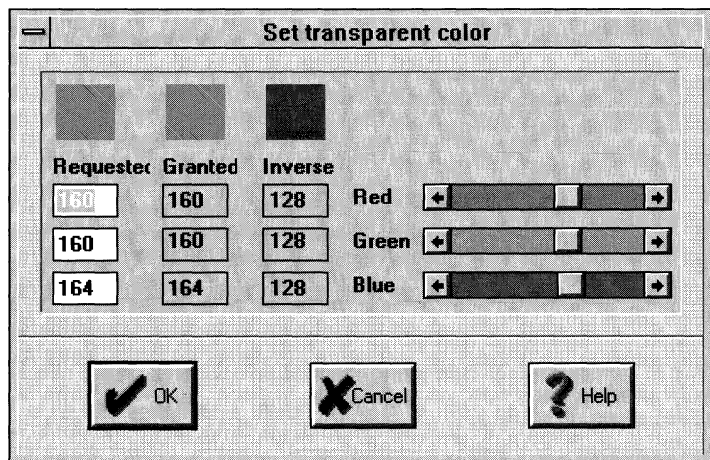
Changing the color of transparent and inverted areas

Changing the color of the transparent and inverted areas is one way to see how the cursor looks when it's displayed over various colors at runtime. A better way is to test the cursor (see "Testing the cursor" later in this chapter).

You can change the colors the Paint editor displays for transparent and inverted areas as follows:

1. Select Transparent or Inverted by doing one of the following:
 - In the Colors palette, double-click with either mouse button on either the Transparent or the Inverted bar.
 - Choose Cursor | Edit Transparent Color.
2. Resource Workshop brings up the following dialog box:

Figure 10.5
Edit Transparent and
Inverted Colors dialog box



This dialog box works similarly to the Edit Colors dialog box (see Figure 8.13 on page 188), except that it changes both the transparent and the inverted color at the same time. You'll notice in this box that there is a second color that's the inverse of the first. The color in this box becomes the color for inverted areas.

Setting the cursor's hot spot

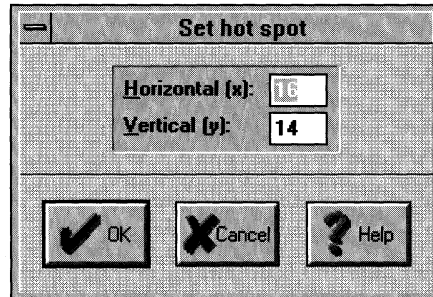
An important consideration when you customize a cursor is where to put the hot spot, the cursor's active area. The hot spot is the single pixel in the cursor that fixes the location when the user places the cursor and clicks to make a selection.

To set a hot spot, decide on the exact pixel coordinates for the hot spot. Pixel coordinates are in horizontal (x) and vertical (y) units. The upper left pixel of the icon image is $x=0$ and $y=0$. The lower right pixel for a 32×32 cursor is $x=31$ and $y=31$, and for a 32×16 cursor is $x=31$ and $y=15$.

Fortunately, you don't have to count pixels. Just place any paint tool (such as the pen) over the exact location where you want the hot spot. The status line at the bottom of the edit window shows you the pixel coordinates. (You should make a note of these coordinates so you can enter them later in the Set Hot Spot dialog box.)

Once you've determined the coordinates, choose **Cursor | Set Hot Spot**.

Figure 10.6
The Set Hot Spot dialog box



The following instructions explain in detail how to set the hot spot:

1. With the cursor in the Paint editor, zoom in on the cursor image until it's big enough to let you precisely choose the pixel coordinates for the hot spot.
2. Make sure the grid is displayed on the zoomed image. If necessary, choose **Options | Editor Options** and check **Grid On Zoomed Windows**.

3. Select a paint tool that lets you precisely point to a pixel. The Line tool is a good choice because it includes a cross hair to show exactly where its hot spot is.
4. With the paint tool you've chosen, point to the location on the zoomed image where you want the hot spot and look at the coordinates displayed on the status line. Make a note of these coordinates.
5. Choose Cursor | Set Hot Spot and type the coordinates you read earlier.

Testing the cursor

Anytime you want, you can test your cursor by choosing Cursor | Test.

Resource Workshop turns the current cursor into a test version of your cursor. You can move it anywhere to see how it looks on different color backgrounds. When you finish testing, just click the mouse to get back to a real cursor.

To test the cursor's hot spot,

1. Zoom the cursor image with the grid turned on.
2. Select a paint tool with a cross hair, like the Paint Can tool.
3. Choose Cursor | Test.
4. Move the hot spot to a particular pixel on the zoomed image and click the mouse. When you click the mouse, the test cursor disappears and is replaced by the paint tool you selected in step 2. If you correctly set the hot spot, the paint tool points to the same pixel your test cursor pointed to.

Saving changes

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. The primary method for saving changes you've made to a cursor resource is to save the entire project. You can also save the cursor resource separately as a file, but you use this option only under special circumstances.

Saving a project

This is the Save option you'll use most often.

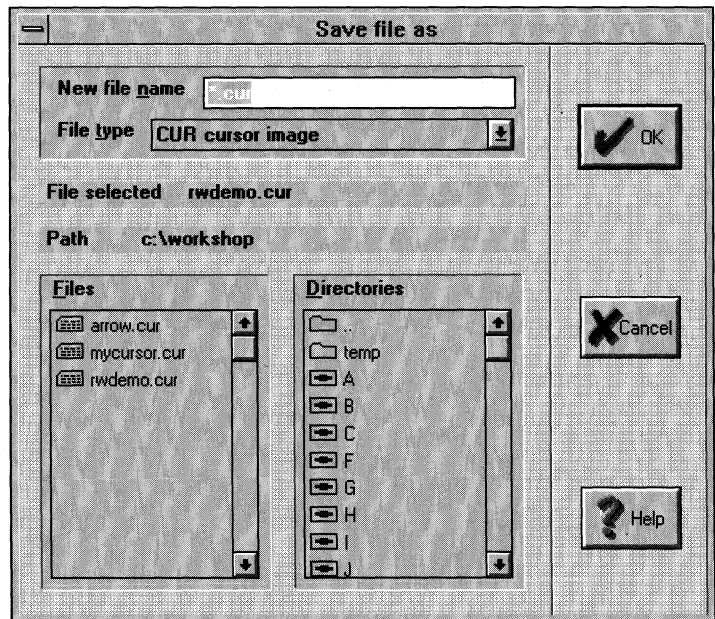
To save an entire project, choose File | Save Project. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. Any changed resource linked to an external file is updated in the external file.

Saving a cursor resource as a file

Saving an entire project also saves the cursor resource you're working on. However, you might want to save the cursor resource separately as a file in order to put it in binary format for use by other programs. In this case, you choose Resource | Save Resource As.

1. Choose Resource | Save Resource As. Resource Workshop displays the Save Resource As dialog box.

Figure 10.7
Save File As dialog box for a
cursor resource



2. Either enter a new file name or choose from the Files list the name of a file you want to overwrite. If you want to put the file in another directory, you can either change the path by using the Directories list or enter the path when you type the

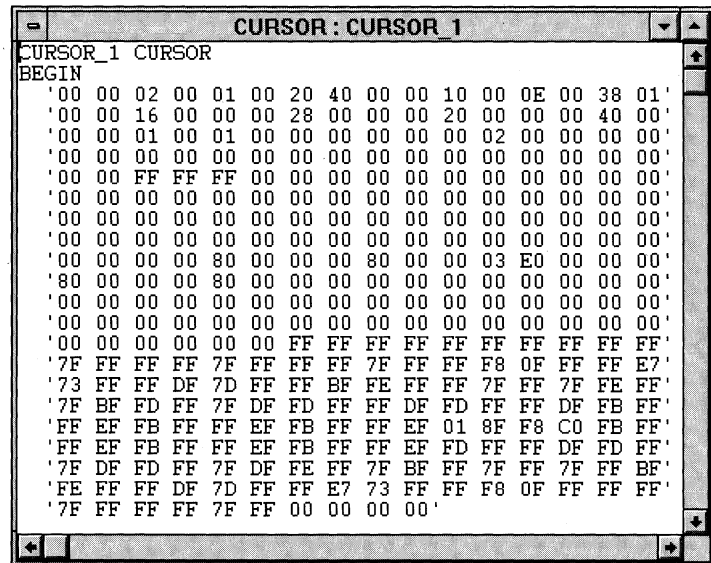
file name. When you're satisfied that the file name is correct, press *Enter* or click OK.

3. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the cursor to be saved in binary format to the external cursor file and not in the project file or in any previous cursor file.

Editing a cursor resource script

If you wish, you can edit the resource script of a cursor by selecting the cursor resource in the project window and choosing Resource | Edit as Text. Resource Workshop brings the resource script up in an edit window.

Figure 10.8
Cursor resource script in Edit
window



```
CURSOR : CURSOR_1
CURSOR_1 CURSOR
BEGIN
'00 00 02 00 01 00 20 40 00 00 10 00 0E 00 38 01'
'00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
'00 00 01 00 01 00 00 00 00 00 00 02 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 80 00 00 00 80 00 00 03 E0 00 00 00'
'80 00 00 00 80 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF'
'7F FF FF FF 7F FF FF FF 7F FF FF F8 0F FF FF E7'
'73 FF FF DF 7D FF FF BF FE FF FF 7F FF 7F FE FF'
'7F BF FD FF 7F DF FD FF FF DF FD FF FF DF FB FF'
'FF EF FB FF FF EF FB FF FF EF 01 8F F8 C0 FB FF'
'FF EF FB FF FF EF FB FF FF EF FD FF FF DF FD FF'
'7F DF FD FF 7F DF FE FF 7F BF FF 7F FF 7F FF BF'
'FE FF FF DF 7D FF FF E7 73 FF FF F8 0F FF FF FF'
'7F FF FF FF 7F FF 00 00 00 00'
```

The only readily comprehensible parts of a cursor resource script are the first and second lines ("BEGIN" and the resource name) and the last line ("END"). Everything between the second line and the last line is hexadecimal code. If you like, you can edit this code to see the effect on the cursor, but do so at your own risk.

Creating bitmaps

Bitmap resources display graphic images in your Windows program. Windows uses bitmaps to represent scroll bar arrows, the Minimize and Maximize buttons, and so on. You can also use Bitmap resources to create a brush pattern that your program can use to fill a display area.

Figure 11.1
The Brush bitmap from the
Paint Editor tool box



To design your bitmaps, use Resource Workshop's versatile Paint editor, which includes numerous tools and a color palette. In the Paint editor, you can look at and zoom different views of the bitmap you're creating.

Drawing a bitmap is very similar to drawing an icon, except you can't add transparent or inverted areas to a bitmap image. You can, however, change the size and attributes of your bitmap image with the Size and Attributes commands on the Bitmap menu.

If you're ready to start and need help in creating your bitmaps, we recommend you read the following:

- Chapter 8, "Using the Paint editor," which introduces all the Paint editor functions. Most Paint editor functions are the same for designing icons, cursors, bitmaps, and fonts. Chapters 8 through 12 give you detailed information about any differences.

- “Creating a sample icon” in Chapter 9. This includes a step-by-step tutorial for creating icons, and shows you some of the Paint editor features.

When working with bitmaps, you perform four primary tasks:

- Starting the Paint editor
- Customizing a bitmap
- Saving the bitmap
- Testing the bitmap

Starting the Paint editor

To work on a bitmap, you need to start the Paint editor by creating a new bitmap or choosing one that already exists.

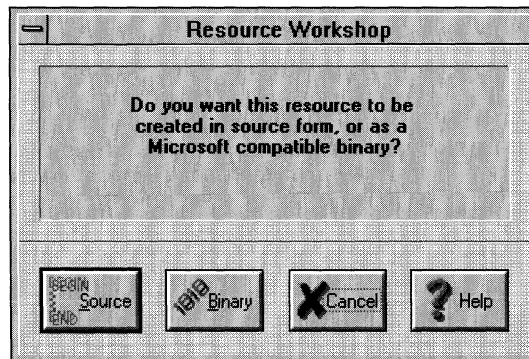
Creating a new bitmap

See Chapter 3 if you need information about opening a project.

To create a new Bitmap resource in the project window,

1. Open an existing project or create a new one.
2. Choose Resource | New and select the .BMP resource type. Resource Workshop asks whether you want to create a bitmap in source form or binary format.

Figure 11.2
Dialog box prompting for
source or binary

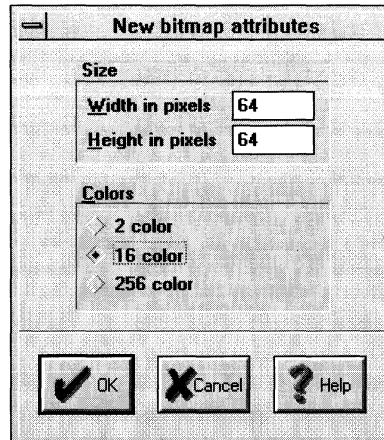


Creating your bitmap in source form consolidates resources because all source resources can be stored in your script file. If your bitmaps need to be compatible with Microsoft compilers, or if you want to share the same bitmap across several projects, store your bitmaps in binary format (.BMP files).

3. Choose Source to store the bitmap as resource script, or choose Binary to store the bitmap as binary format.

If you choose Source, the New Bitmap Attributes dialog box appears.

Figure 11.3
The New Bitmap Attributes
dialog box



These options let you determine

- The bitmap image size. Type in the width and height of the bitmap image you want to create in pixels.
- The number of colors you want to work with while designing your bitmap. Choose 2 color, 16 color, or 256 colors. The capabilities of your computer system may limit your choices. For example, if your computer can't display 256 colors, that option will be dimmed and you won't be able to select it.

After choosing the options you want, choose OK, and the Paint editor appears.

If you choose Binary, Resource Workshop displays a dialog box that lets you name the bitmap file and indicate where you want to store the resource. When you're through in the dialog box, choose OK, and the Paint editor appears.

*See page 201 for more
information on creating a
binary resource.*

Editing existing bitmaps

After you've opened the project containing the existing bitmap, find the BITMAP entry you want to edit in the project window. Each BITMAP entry in the project window represents one Bitmap

resource. Double-click the name, or select the name and choose Resource | Edit.

You'll see the Paint editor for your bitmap.

If you want to edit the resource script of a bitmap, select the bitmap from the project window by clicking it, and then choose Resource | Edit As Text.

Customizing a bitmap

*Chapter 8 describes the
Paint editor tools.*

Once you've displayed a bitmap image in the Paint editor, use the paint tools to make the changes you want. For example, you can draw new portions of the bitmap, or, if you're changing an existing bitmap, erase some of the image. Many procedures you learned in working with other bitmapped resources apply to bitmaps also. You won't, however, be able to add transparent or inverted areas to a bitmap image.

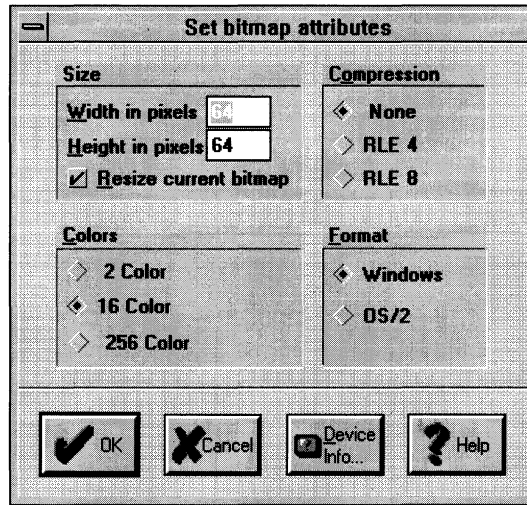
Changing bitmap size and colors

The Bitmap | Size and Attributes commands let you change the following:

- The bitmap image size and whether the image you've already drawn will shrink or stretch if you change the overall image size.
- The number of colors used in the image.
- How the bitmap should be stored (in a compressed or uncompressed format and in a format compatible with Windows or OS/2).

To change any of these attributes for the bitmap image you're working with in the Paint editor, choose Bitmap | Size and Attributes. You'll see the Set Bitmap Attributes dialog box.

Figure 11.4
The Set Bitmap Attributes dialog box



Here are your options:

Table 11.1
Bitmap size and attribute
options

Option	Description
Size	Type the width and height, in pixels, of the total bitmap image. The maximum size you can work with is limited by the amount of available memory on your computer. Even though you could type 9999 for both the width and height of your image, it's likely that the maximum size for bitmaps on your computer is far less. Check the Resize Current Bitmap option if you want an image that's already drawn to stretch or shrink when you resize your bitmap. For example, suppose you create a 64×64 pixel bitmap. If you want the same image to fit into an area that's only half the size (32×32 pixels), check Resize current bitmap when you specify the new size.
Colors	If your computer can work with colors, you can choose to create 16-color and 256-color bitmaps. Any color options that your computer can't handle are dimmed, meaning you can't choose them. So if your computer can create 16-color images but can't create 256-color images, you'll only be able to choose 2 or 16 colors; the 256 color option will be disabled.

Table 11.1: Bitmap size and attribute options (continued)

Option	Description
Compression	Resource Workshop lets you compress your bitmap images.
None	Choose this option for 2-color bitmaps.
RLE4	Choose this option for 16-color bitmaps.
RLE8	Choose this option for 256-color bitmaps. (RLE stands for "run length encoded.") You may want to experiment with both compressed and non-compressed color images because sometimes the compressed image takes up more space than the uncompressed image.
Format	Lets you store the bitmap in Windows or OS/2 format.

Saving a bitmap

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. There are two options you can use to save changes you've made to a Bitmap resource, although you'll seldom need the second method. You can

- Save the entire project
- Save the Bitmap resource as a file

Saving the project

To save the entire project, choose File | Save All. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. It also updates any changed resource linked to an external file. This is the Save option you will use most often.

Saving the bitmap as a file

Saving the entire project also saves the bitmap you're working on. If you want to save only the bitmap, you can choose Resource | Save Resource As to save the bitmap as a file. The only time you're likely to do this is if you want to put the bitmap in binary format in a separate file for the first time. To save the resource as a file,

1. Choose Resource | Save As. Resource Workshop displays the Save File As dialog box.
2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or enter the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or choose OK.
3. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. If you choose Yes, all future changes to the bitmap are saved in binary format to the external bitmap file and not in the project file or in any previous bitmap file.

Testing the bitmap

For information about compiling resources into an executable file, see Chapter 3.

To test your bitmap, you'll need to compile the Bitmap resource and bind it to an executable (.EXE) file. Then you can run the executable file to see what the bitmap looks like.

Creating fonts

A font is a collection of data used by a computer to draw or “realize” individual characters on an output device such as a display monitor or a printer. The font contains data that describes the overall collection, such as the typeface name, the suggested size, the character set, the letters in the font, and so on. The font also contains the information the computer needs to draw each character.

Windows supports two basic types of fonts: raster fonts and outline fonts. Raster fonts contain a bitmapped image of each character. Outline fonts contain a series of drawing commands for each character. Usually, they also contain “hints” or “fudge factors” that the computer uses to produce better quality images at various sizes. The outline fonts that Windows supports lack these hints. These limited outline fonts are called vector fonts, of which Roman, Script, and Modern are examples. As a Windows user, more sophisticated font technology is available to you through third-party font rasterizers.

Resource Workshop creates and edits Windows raster fonts only.

Although you can use Resource Workshop to create customized letters, you’ll probably want to use a specialized font development package to do that kind of work. The kind of fonts you’re more likely to use Resource Workshop for are picture fonts: small bitmaps that you want to group together.

If picture fonts are bitmapped images, why wouldn’t you create them just as you would a bitmap? In some cases, font resources

and bitmap resources aren't interchangeable. If you're creating a brush to paint an area onscreen, you'll need to use bitmap resources. But for creating bitmap images, such as a bomb or a stop-sign bitmap image to display onscreen, you could use either bitmap or font resources. For example, this bomb image was created as a font resource.

Figure 12.1
A customized bomb
character



There are a couple of reasons why you might want to define images as part of a font resource instead of bitmap images:

- It's simpler to write Windows code to load a font into memory and paint it. Loading and painting the same image that's stored as a bitmap is more complex.
- If efficient memory use is important, and you're trying to decide whether to create a single image as a font resource or as a bitmap resource, you should probably define it as a bitmap. That's because a font resource has a certain amount of memory overhead; each time you load a font into memory, Windows also loads the font header (see page 249).
- A font can contain a large number of images, and for multiple images, it may be more efficient to store them as part of a single font resource. When you need a series of bitmap images, you can create these images as part of a single font resource. Then at runtime, you'll only need to load a single resource.

When working with a font resource, you'll perform five primary tasks:

- Starting the Paint editor
- Customizing a font resource
- Saving the font resource
- Adding a font resource to your application
- Testing the font resource

The first task, starting the Paint editor, displays a collection of font images ready for you to customize. The second task, customizing a font image, is a function of the Paint editor itself. The third task, saving the font resource, happens automatically if you want the font resource stored in your project file. If you want to save a font resource in a separate file, you'll have a few steps to follow. The

fourth and fifth tasks, adding a font resource to an application and testing it, are done outside of Resource Workshop.

Starting the Paint editor

To work with a font resource, you need to start the Paint editor by modifying one that already exists or creating a new one.

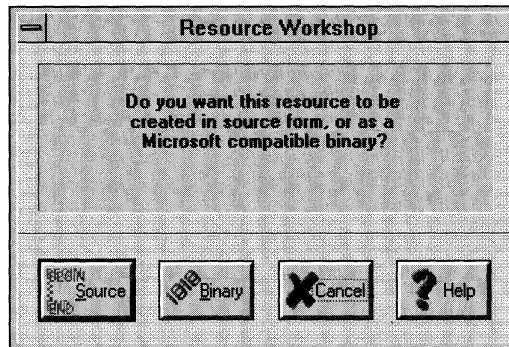
Creating a new font resource

See Chapter 3 if you need help opening a project.

To open the Paint editor to create a new font resource,

1. Open an existing project or create a new one.
2. Choose Resource | New and select the .FNT option. Resource Workshop asks whether you want to create a font in source form or binary format.

Figure 12.2
Dialog box prompting for
source or binary



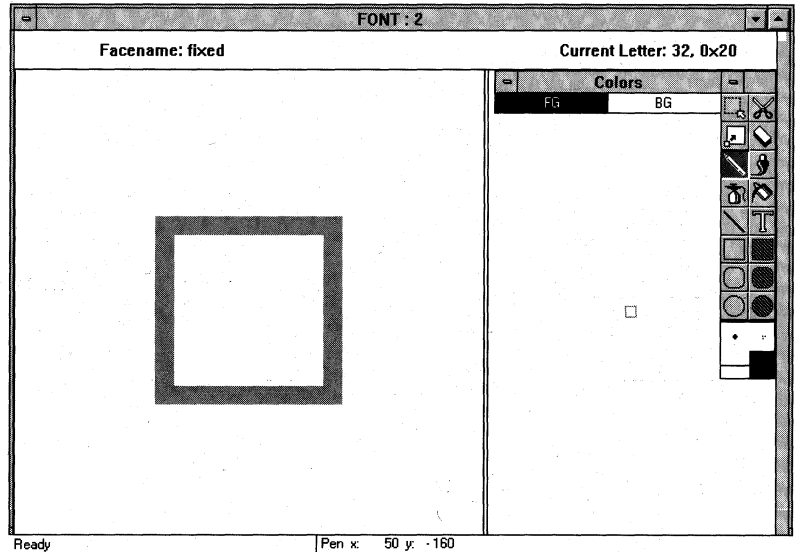
Saving your font as resource script consolidates resources because all source resources can be stored in your project file. But if you need your fonts to be compatible with Microsoft compilers, or if you want to share the same font across several projects, store your fonts in binary format in .FNT files.

3. Choose Source to store the font as resource script, or choose Binary to store the font as binary format.

If you choose Source, you'll see the Paint editor. If you choose Binary, Resource Workshop displays a dialog box that lets you name the binary font file where you want to store the resource. Type in a file name, choose OK, and the Paint editor appears.

See page 201 for more information on creating a binary resource.

Figure 12.3
The Paint editor as it looks for
a new font



Editing an existing font resource

*See Chapter 3 if you need
help opening a project.*

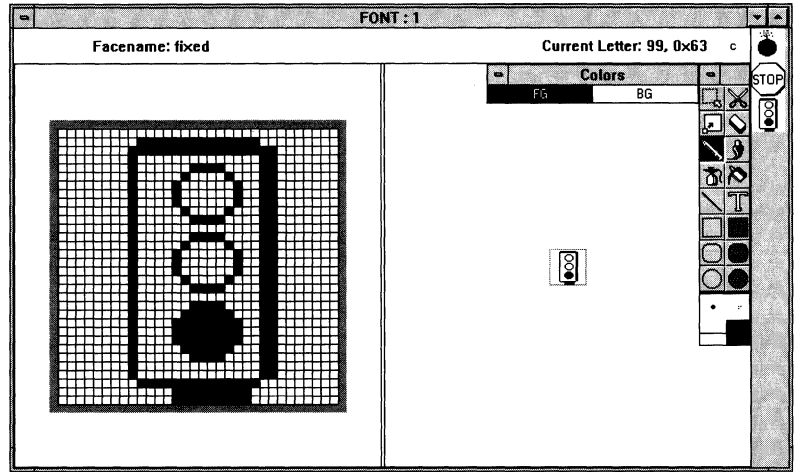
To open the Paint editor and edit an existing font resource, open the appropriate project. Resource Workshop displays the Project window.

Each FONT entry in the project window represents one font resource, though a single font can contain many images. Double-click the font resource in the project file, or select the name and choose Resource | Edit to load the font into the Paint editor.

Customizing a font resource

After you have the font resource open in the Paint editor, you can begin working on it. You can use all the Paint editor features described in Chapter 8. The Paint editor makes it easy to work with all the characters or images in a font resource. While the left side of the Paint editor displays paint tools—just as it does when you're editing an icon, cursor, or bitmap—the right side of the Paint editor displays all the characters or images in the font you're editing.

Figure 12.4
Font characters in the Paint
editor



By default, the Paint editor loads the font image at the top of the display. You can click on any of the font images on the right side of the Paint editor to load one in and work on it.

Changing a font image

Once you've displayed a font image in the Paint editor, use the paint tools to make the changes you want. For example, you can draw new portions of the font or erase some of the existing image.

If you want to work with the resource script of a font, select the font from the project window by clicking it, and then choose **Resource | Edit As Text**.

Defining the character set for a font

When you create a new font resource, it includes only one 8x8 pixel image. Usually you'll want more than one image in your font resource. You also might want to specify a different size for your font images. To specify more than one image in a font resource and to change the size of font images, use the **Font | Font Size** option.

To define the font size,

1. With the font displayed in the Paint editor, choose **Font | Font Size** to specify the font size. You see the **Font Size Information** dialog box.

Figure 12.5
The Font Size Information
dialog box

2. Make your choices. Your images in a font resource can vary in width or they can be all the same size. Use these Font Size options to choose the size for your font images.

Table 12.1
Font size options

Option	Description
Width	If you want all images to be the same width, type the width in pixels. If you want the widths to vary, type a 0 (zero) here and specify a maximum width. See page 247 for more on variable-width characters and images.
Height	Type the height, in pixels, of the font images.
Average Width	Resource Workshop calculates an average width for your font images if you have specified 0 for Width (which makes your font images a variable width). Otherwise, Average Width is the same as Width. The Average Width is calculated when you open this dialog box. (So you won't see this value change as you type other changes into this dialog box.)
Maximum Width	For variable-width fonts, specify the maximum width in pixels. This option is available only if you have typed 0 next to Width.
Stretch current chars	Check this option if you want existing images to stretch or shrink, based on height and width changes you type into this dialog box.

The Font Size Information dialog box also contains character options that let you choose how many images to include in your font resource. Decide how many images you want in your font resource. (You can always change your mind later.)

Choose a range of decimal codes to map your font images to the ANSI character set. For example, to map a font image to the character *a*, specify the decimal code 97. The image itself needn't be the character *a*, unless you want it to be. In the sample font resource discussed at the end of this chapter, a bomb image is mapped to *a*. The second font image would then be mapped to the character *b*, decimal code 98.

Use these Character options to map the character set or images to be included in your font resource:

Table 12.2
Character options

Option	Description
First	Type an ANSI decimal code to define the first image in your font. For example, if you want the first image to correspond to <i>a</i> , type 97.
Last	Type an ANSI decimal code to define the last image in your font. For example, if you want the last image to correspond to <i>z</i> , type 122.
Default	Type an ANSI decimal code to define the default font image that will be displayed when you edit this font resource. The Default value must be within the character range defined by the First and Last values. For example, if you've typed 97 for First and 122 for Last, you can't type 88 for Default.
Break	Type an ANSI decimal code to define a break character for your font resource. The Break value must be within the character range defined by the First and Last values.

Creating variable-width fonts

If you're creating a character set, you need to decide whether you want all your letters to be the same width (like this) or in a variable width (like this). Nonvariable-width fonts such as Courier, Elite, and Pica look like typewriter-style letters. Variable-width fonts such as Helvetica, Times Roman, and Palatino assign different widths to the letters. For example, this manual uses Palatino, a variable-width font; therefore, the letter *i* is not as wide as the letter *w*.

If you're using a font resource to define a picture font that isn't really based on a character set, you may or may not want all the images to be the same width.

While you're editing a font, you can choose to define variable widths for images by choosing Font | Font Size.

Next to Width, type 0 (a zero); next to Maximum Width, type the maximum width, in pixels, for the images in the font resource.

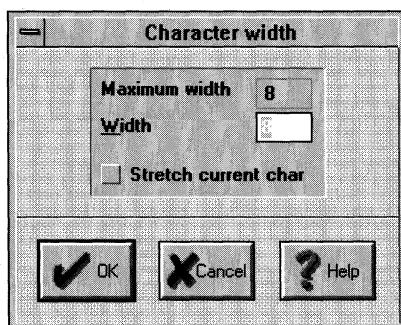
Once you've chosen a width of zero in the Font Size Information dialog box, you can choose the width for an individual image using the Font | Character Width command.

For example, suppose you've used the Font | Font Size option to define the following sizes for your font resource:

- Width = 0
- Height = 32
- Maximum Width = 32

Based on these sizes, all images will be 32 pixels high, and each image can vary in width, up to a maximum of 32 pixels. As you're editing a particular image, you can use the Font | Character Width option to define the width of that image. You'll see the Character Width dialog box.

Figure 12.6
The Character Width dialog
box



Next to Width, type a value that's less than or equal to the Maximum Width. In addition, you can check the Stretch current char option if you want the existing image to stretch or shrink based on the width change you type into this dialog box.

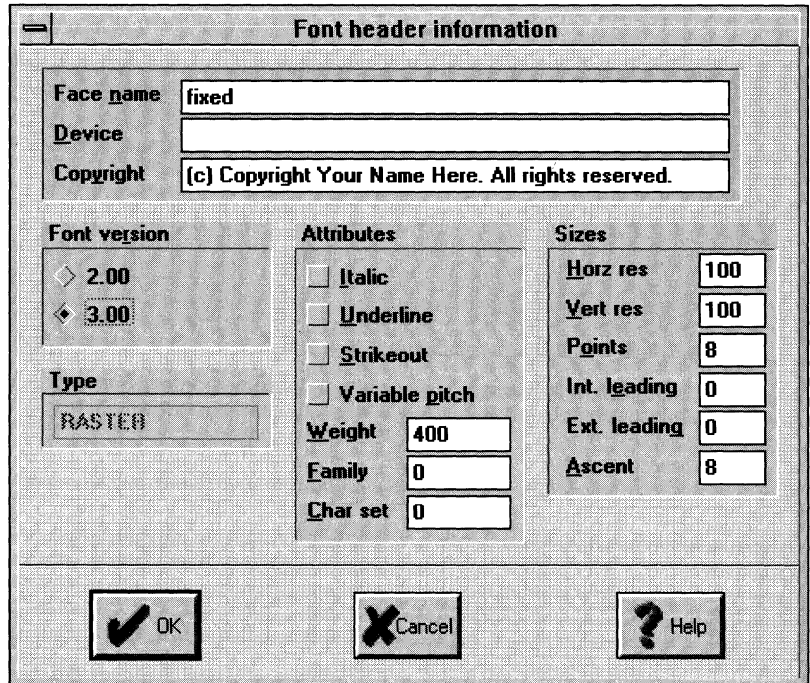
Defining a header for a font resource

Every font resource includes a header that describes general information about the font, such as typeface name and copyright information. If you're defining a font resource that consists of alphabetic characters, the header defines typestyle and size for all characters in the font.

To define the header for a font,

1. Display the font in the Paint editor by double-clicking the font name in the project window, or by using Resource | New to create a new font resource.
2. In the Paint editor, choose Font | Header to specify header information. You'll see the Font Header Information dialog box.

Figure 12.7
The Font Header Information dialog box



The dialog box is titled "Font header information". It contains several fields and sections for configuring font metadata.

Font header information	
Face name	fixed
Device	
Copyright	(c) Copyright Your Name Here. All rights reserved.
Font version	
◆ 2.00	
◆ 3.00	
Type	
RASTER	
Attributes	
☐ Italic	
☐ Underline	
☐ Strikeout	
☐ Variable pitch	
Weight 400	
Family 0	
Char set 0	
Sizes	
Horz res 100	
Vert res 100	
Points 8	
Int. leading 0	
Ext. leading 0	
Ascent 8	

At the bottom, there are three buttons: OK (with a checkmark icon), Cancel (with an X icon), and Help (with a question mark icon).

3. Define the header for your customized fonts. Here are your choices:

Table 12.3
Font header options

Option	Description
Font Version	Choose Windows 2.00 or 3.00 font version. For reliability, you should always choose 2.00.
Face Name	Type a name that you want to assign to your font.
Device	Type a device name for your font if you want to inform your programs that this font can be used only on a particular device.
Copyright	Type copyright information for your custom font.

Use the attribute options to tell Windows and your applications that the font itself implements these attributes:

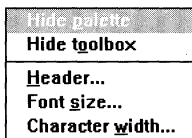
Table 12.4
Font attributes

Option	Description
Italics	The font contains italicized characters.
Underline	The font contains underlined characters.
Strikeout	The font contains characters that are struck out.
Variable Pitch	The font is a variable pitch font.
Weight	The font is of normal weight (400) or boldfaced (700).
Family	Describes the font family. The acceptable values are: 0 Don't care 1 Roman 2 Swiss 3 Modern 4 Script 5 Decorative
Char set	Defines the character set. The value can be 0 – 255, but only 0, 2, and 255 have predefined meanings: 0 ANSI, the default Windows character set 2 Symbol, used for math and scientific formulas 255 OEM, a machine-specific character set

Table 12.5
Font sizes

Option	Description
Horizontal Res.	Horizontal number of pixels per logical inch on your video display.
Vertical Res.	Vertical number of pixels per logical inch on your video display.
Points	Type size. A point is 1/72 of an inch. A character is measured from the top of the ascender to the bottom of the descender. The value you enter here should not include space for diacritics.
Internal Leading	The space reserved for diacritics in pixels.
External Leading	The additional space between lines of characters in pixels.
Ascent	The height of the character above the baseline in pixels.

Changing size and attributes



When you're editing a font resource in the Paint editor, the Font menu has three commands that let you make changes to the font images.

- **Header** defines the header information for your font resource, including the font version, the font name, copyright information, and so on. For more information, see page 249.
- **Font Size** defines the character set in this font, and the width and height of each character. For more information, see page 245.
- **Character Width** specifies the width for a particular image in a variable width font resource. (This command won't be available unless you've already defined a variable width font using Font | Font Size.) For more information, see page 247.

Deleting a font image

To delete a font image,

1. Double-click the font image you want to delete so it is displayed in the Paint editor.
2. Choose Edit | Select All to select the font image.
3. Choose Edit | Cut, Edit | Delete, or press *Del* to delete the image. The image disappears. The image may still appear on the right

side of the Paint editor, but will disappear when you select another image to edit.

4. Choose File | Save All to save your change.

Saving a font resource

It's a good idea to save changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. There are two primary options you can use to save changes you've made to a font resource. You can

- Save the entire project
- Save the font resource as a file

Saving the project

To save the entire project, choose File | Save all. Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. Any changed resource linked to an external file will be updated in the external file. This is the Save option you'll use most often.

Saving a font resource as a file

Saving the entire project also saves the font resource you're working on. If, however, you want to save only the font resource, you can choose Resource | Save As to save the font resource as a file. Because this step is more complicated than just saving the project, you aren't likely to do it unless you want to put the font resource in binary format in a separate file for the first time. To save a font resource as a file,

1. Choose Resource | Save As. Resource Workshop displays the Save File As dialog box.
2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or enter the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or choose OK.
3. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Choosing

Yes causes all future changes to the font resource to be saved in binary format to the external font file and not in the project file or in any previous font resource file.

Adding font resources to your application

You must create .FON files outside of Resource Workshop.

Although Resource Workshop stores your fonts in your project file or in a .FNT file, when you add the font resource to your application, you'll also need to create a .FON file. You can do this using Turbo Pascal or C++.

Creating a .FON file with Turbo Pascal

Before you create a .FON file, save the project file as a resource object (.RES) file. See page 56 if you need more information about resource object files.

To create a .FON file for the RWPDEMO sample font (see page 254),

1. Write and compile this Pascal program to create a resource-only dynamic-link library.

```
Library Stoppers;
{$D Stoppers 133,96,72: Bomb, Stop sign, Stoplight}
{$M 1024,0}
{$R Stoppers.res}
begin
end.
```

2. At the DOS command line, rename STOPPERS.DLL to STOPPERS.FON:

```
rename STOPPERS.DLL STOPPERS.FON
```

Creating a .FON file with C++

Before you create a .FON file, save a project file as a resource object (.RES) file. See page 56 if you need more information about resource object files.

To create a .FON file for the RWCDEMO sample program (see page 254),

1. Write and compile this C++ program:

```
#include "windows.h"
#pragma warn -par      // Don't warn about unused parameters
int FAR PASCAL Lib (HANDLE hInstance, WORD wDataSeg, WORD wHeapSize,
                    LPSTR lpszCmdLine)
{
    if (wHeapSize > 0)
        UnlockData(0);
    return 1;
}

#pragma war .par      // Warn about unused parameters
```

2. Create a module definition file.

```
LIBRARY Stoppers
DESCRIPTION 'STOPPERS 133,96,72: Bomb, Stop sign, Stoplight'
STUB 'WINSTUB.EXE'
DATA NONE
```

3. Create a make file and execute it.

```
stoppers.dll: stoppers.cpp stoppers.res stoppers.def
    bcc -mc! -WD stoppers.cpp
    rc stoppers.dll
    ren stoppers.dll stoppers.fon
```

Testing the font

To test your font, you'll need to compile the font resource and bind it to an executable file. You can then run the executable file to see what the font looks like.

For information about binding resources to a executable file, see Chapter 3.

A sample font resource

To see how you can use Resource Workshop to create a font resource, take a look at a sample font resource:

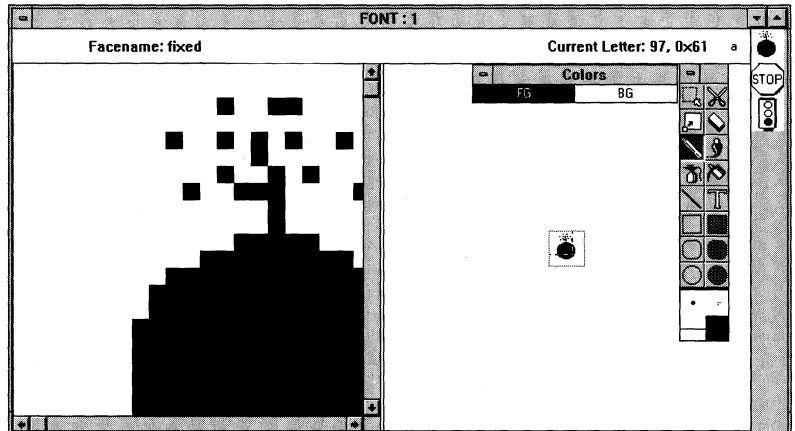
1. Open the RWPDEMO.RC project file that came with Resource Workshop.
2. In the project window, find the font resource entry called 1.

3. Open the resource either by double-clicking it or by selecting it and choosing Resource | Edit. In the Paint editor, you'll see a bitmap that looks like a bomb.

If you've already used the Paint editor to create icons, cursors, or bitmaps, you'll recognize most of what you see. On the left side, the tool box includes all the paint tools you can use to draw any bitmap image. The way the image itself looks, and the status line at the bottom of the Paint editor are also the same.

There are some additional Paint editor features that will help you edit your font resources. Notice the status line across the top of the image that tells you about the font's typeface (on the left) and the current character you're editing (on the right). And, on the right side of the Paint editor, there is another, smaller version of the bomb image, along with stop-sign and traffic light images.

Figure 12.8
RWPDEMO font in the Paint
editor



The right side of the font status line shows which character you're editing: "Current letter: 97, 0x61 a." Of course, you aren't really editing the letter *a*, but the bomb bitmap you're looking at is mapped to the letter *a*, whose ANSI identifier is decimal 97 or hexadecimal 61.

The list of small bitmaps along the right side of the editor shows you all the characters or images that are defined for this font resource: the bomb, the stop sign, and the stop light. Although it looks as if the Paint editor wouldn't be able to fit many more bitmap images down the right side of the screen, you can define many images in your font resource. If there are more images than can fit down the right side of the Paint editor, Resource Workshop displays scroll buttons so you can browse through the entire list.

Look at another character. Click the stop-sign bitmap image. Notice the way the font status line changes to show you that now you're editing the bitmap mapped to *b*.

Now look at the dialog boxes that were used to define this font resource. You can open these dialog boxes using two commands in the Font menu, Header and Font Size.

1. Choose Font | Header and you'll see the dialog box you use to define the header for this font resource.

Every font resource includes a header to describe general information about the font. Note, for example, that the copyright information indicates that this font resource belongs to Borland International.

2. Choose OK or Cancel to close the Font Header Information dialog box.

3. Choose Font | Font Size and you'll see the dialog box you can use to define the size of the images in this resource, as well as the number of images in the resource.

The Size option in this dialog box shows that the font images are 32×32 pixels. You can define images that vary in width by entering a zero next to width and then entering a maximum width (see page 245).

The Character option in this dialog box shows the ANSI codes of characters included in this font. This font resource includes characters whose ANSI decimal codes are 97, 98, and 99 (or the characters *a*, *b*, and *c*). You can also define a default and a break character for your font (see page 245).

4. Before you close the Font Size Information dialog box, change Last from 99 to 199. This shows you what the Paint editor looks like when there are more images in the font than can fit on the right side of the screen.
5. Choose OK to close the Font Size Information dialog box. You'll see empty images below the stop-light bitmap. Note the scroll buttons above and below the list of images. You use these buttons to scroll through all the images that are defined for this resource.

Click any image that you want to edit on the right side of the Paint editor. You can choose an existing image, or choose any of the empty boxes below the stop light to add a new image.

Creating user-defined resources

In addition to the resource types discussed in previous chapters, you can also define your own resources. After you create a new resource type, you can add any number of *user-defined resources* of this type to your project, just as you can add resources of any of the standard, predefined types (such as dialog windows, menus, and bitmaps).

So why would you want to define your own resource types? Well, user-defined resources can contain data that doesn't fit into one of the standard resource types (dialog boxes, menus, accelerators, and so on). For example, if you want to create a character string resource that's longer than the STRINGTABLE limit of 255 characters, you can define your own resource type and store your character strings there.

See Charles Petzold's *Programming Windows for more information on metafiles*. (See page 6 for the bibliography listing for this book.)

You can also include *metafiles* in your project as user-defined resources. A metafile is a type of bitmap (in source form, it's a collection of Graphics Device Interface (GDI) calls) that's not only easier to scale and more device-independent than the standard Bitmap resource, but also often takes up less storage space than a Bitmap resource.

When you define a new resource, you can store the data either as part of the resource definition in a project file or as a separate file. As with any resource, Resource Workshop can compile the data and bind it to your executable file to make the data available to your application at runtime.

- You can also use the RCDATA resource type to add data to your application. See page 262 for more information.

When working with user-defined resources, you perform five primary tasks:

- Creating a user-defined resource type
- Adding a user-defined resource to the resource type
- Editing a user-defined resource
- Testing the user-defined resource
- Saving the user-defined resource

Creating a resource type

Before you can add a user-defined resource to your project, you must first create a type for it, as follows:

1. Make sure you've already opened a project. (See Chapter 3 if you need information about opening a project.)
2. Choose Resource | New.
3. In the New Resource dialog box, press the New Type button.
4. In the New Resource Type dialog box, type a name for the resource type you're creating. For example, if you're creating a resource to contain a large block of text, you could name your new resource type TEXT.
5. When asked if you want to create an identifier for the new resource type, click Yes, then enter a value for the identifier in the New Identifier dialog box. This value is the ID that Windows and your program will associate with this identifier type.

If you use a numeric ID, it must be greater than 255, because Windows reserves the values 1 through 255 for standard resources.

Once you've defined a new resource type, whenever you create a new resource, you'll see that resource type listed in the New Resource dialog box with all the standard resource types (ACCELERATOR, BITMAP, USER-DEFINED RESOURCE, and so on).

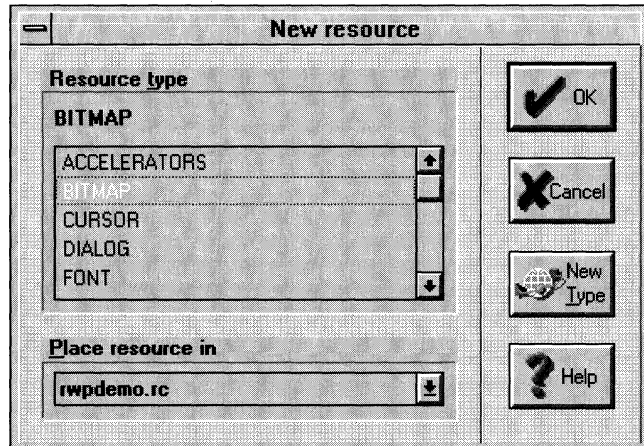
Adding a user-defined resource

After you've created a resource type, you can add a resource of that type to your project, as follows:

1. Make sure you've already opened a project. (See Chapter 3 if you need information about opening a project.)
2. Choose Resource | New. Resource Workshop displays the New dialog box with Resource already checked. Click OK to create a new resource.
3. In the New Resource dialog box, choose your user-defined resource type. Under "Place resource in", choose the project file (probably the current one) where you want to store the resource.

If you don't see the resource type you want, press the New Type button to define a new type.

Figure 13.1
The New Resource dialog box



4. After you choose OK in the New Resource dialog box, Resource Workshop opens the text editor with a blank definition for your user-defined resource. For example, if you add a user-defined resource called TEXT, you see the following code in the text editor:

```
TEXT_1 TEXT
BEGIN
END
```

At this point, you can edit the resource, as described in the next section.

Editing a user-defined resource

See Chapter 3 for help with opening a project.

To edit a user-defined resource, you must have a project open. You can either create the resource (see the previous section) or you can open an already existing one in the Project window.

The resource name is listed in the Project window by its user-defined resource type, just as any resource of a predefined type is listed. For example, if you create a TEXT resource type and a Text resource, you see a TEXT resource entry in the Project window.

To open the editor,

- Double-click the name of the resource you want to edit.
- Select the resource name, then choose Resource | Edit or Resource | Edit As Text.

Once you've brought the resource up in the text editor, you can add data to it or edit the data in it. Here's an example of what you'll see when you add a new user-defined resource of type RESTYPE to your project:

```
RESTYPE_1 RESTYPE
BEGIN
    [data definitions]
END
```

The first line shows the resource name and type. Resource Workshop constructs a default name for a new resource by appending to the resource type an underscore and an integer. For example, the first RESTYPE resource you add to your project becomes RESTYPE_1, the second becomes RESTYPE_2, and so on.

To add data to your resource, you can do one of the following:

- Use the text editor to type data between the BEGIN and END statements.
- Store the data in a separate file and add the file name to the end of the first line of the resource script.

After you make any changes—whether you add data between the BEGIN and END or type a file name on the first line—you must recompile the resource to save your changes. If you exit without recompiling, you lose all your changes.

Entering data in the resource script

See page 41 for a
description of the internal
text editor.

When you use the text editor, *Tab*, *Del*, *Home*, *End*, *PgUp*, *PgDn*, and *Backspace* function as usual. However, don't use this editor to do much formatting because Resource Workshop is likely to rearrange the text when it compiles or decompiles this resource.

Here are some guidelines for specifying data between the **BEGIN** and **END** parameters:

- The data can include any combination of numeric values and strings.
- You can use hexadecimal, octal, or decimal notation to represent numeric values.
 - Use either 0x (a zero followed by the letter *x*) or \$ (a dollar sign) as the leading characters for hexadecimal notation. This notation supports only 16-bit values. If you want to use an odd number of hexadecimal values, use the *hexstring* (described later).
 - Use 0o (a zero followed by the letter *o*) or just 0 (zero) as the leading characters for octal notation.
- You can also represent hexadecimal values by using a *hexstring* of hexadecimal values enclosed in single quotation marks. The compiler ignores any spaces you insert to make the hex codes more readable.

For example, you could represent the ASCII values of the characters in the word *string* as '73 74 72 69 65 67' and the hex number 06E07A as '06E07A'.

- If you include text strings in your resource, you should enclose the strings in quotation marks, like this: "*string*".

Strings aren't automatically null terminated. To terminate a string with a null character, type \0 (backslash zero) at the end of the string.

For example, your resource script with data added could look like the following:

```

RESTYPE_1 RESTYPE
BEGIN
    "This is a string."
    0xFFAA 0o7076 01077
    '54 68 69 73 20 69 73 0D 0A 73 6F 6D 65 20 73 61'
    '6D 70 6C 65 0D 64 61 74 61 2E'
END

```

Handling data stored in a separate file

If the data is stored in a separate file, you must use an external editor to edit the file. To add to the resource script a reference to the data file,

- Following the name of the resource type, type the complete path name of the file.
- Delete the BEGIN and END statements.

The following resource script for the resource RESTYPE_1 indicates that the data is stored in the file C:\RW\MYDATA.TXT:

```
RESTYPE_1 RESTYPE c:\rw\mydata.txt
```

Testing a user-defined resource

Because you design a user-defined resource specifically for your program, there's no way to test it from within Resource Workshop. To test your user-defined resource, use Resource Workshop to compile the project file that contains the resource and bind it to your program. You can then run your program and cause it to use the resource.

For information about compiling resources and binding them to a program, see Chapter 3.

Using the RCDATA resource type

RCDATA resources have been included in Resource Workshop for compatibility purposes.

You can use the predefined RCDATA resource type to add a data resource to your application. It works the same as a user-defined resource type. The primary difference between the two is simply addressability: You might prefer to have many different types of user-defined resources rather than just one type, RCDATA.

If you do use an RCDATA resource, add it to the project by choosing File | New and adding a new resource whose type is RCDATA. You'll see a blank RCDATA definition in the text editor, and you can type the data between the BEGIN and END parameters. Use the same rules for typing data as described for user-defined resources.

Deleting a user-defined resource

To delete a user-defined resource from your Project window, select the entry in the Project window by using the mouse or the arrow keys. Then delete the entry by doing one of the following:

- Press *Del* or choose Edit | Delete to simply delete the resource.
- Choose Edit | Cut to delete the resource from the current project and put it into the Windows Clipboard so you can paste it into another project.

Technical notes

Compiler differences

The Resource Workshop resource compiler is almost completely Microsoft compatible and is significantly enhanced over the Microsoft resource compiler in a number of ways.

The following features are improvements over the Microsoft compiler:

- The Resource Workshop compiler allows text descriptions of bitmapped resources (icons, cursors, bitmaps, and fonts), while the Microsoft compiler does not. The text descriptions are written using the resource script language documented in Resource Workshop's online help system.
- The Resource Workshop compiler supports numeric constant expressions for every numeric field, while the Microsoft compiler doesn't. For example, the following statement is correctly interpreted in Resource Workshop, but causes an error with the Microsoft compiler:

```
20 + 20 - 20 ICON foo.ico
```

Resource Workshop interprets this line as defining an ICON resource with an ID of 20 contained in the file FOO.ICO. The Microsoft compiler would attempt to parse this line as a resource type +, with ID 20 contained in the file 20.

- Resource Workshop has added a new fundamental data type, the *hexstring*. This data type consists of a variable number of

hexadecimal digits that describe data bytes, surrounded by single quotation marks. You can also insert spaces for clarity; the compiler ignores them. This new type makes it easier for users to enter hex data. For example, the following hexstring represents a 5-byte hexadecimal number: '010A0B0c0E'. You could also represent this number as follows: '01 0A 0B 0c 0E'.

- The Resource Workshop compiler supports references to files in RCDATA resources as well as in user-defined resources. Support of file references removes the only distinction between user-defined resources and RCDATA resources. If you use the Microsoft resource compiler to compile an RCDATA resource that contains a file reference, you'll get a syntax error.

There is currently one preprocessor-related feature supported by Microsoft that Resource Workshop lacks: The Resource Workshop compiler does not support complex parameterized **#defines**. For example, the Microsoft compiler would successfully compile the following source code, while Resource Workshop would display the error message "Parameterized macros not supported":

```
#define foo( a, b) 10, b, a

RCDATA
BEGIN
    foo( 1, 3),
    foo( 2, 4),
    foo( 5, 6)
END
```

Dialog boxes as child or overlapped windows

Dialog boxes can be used as child windows or overlapped windows. There are at least three examples in Resource Workshop of dialog boxes that are child windows: the Menu editor dialog box, the Accelerator editor dialog box, and the Identifiers dialog box.

See page 6 for the bibliography listing for this book.

For an example of a dialog box used as an overlapped window, see *Programming Windows* by Charles Petzold. The chapter on dialog boxes has an example called Hexcalc that creates such a dialog box.

32x32 16 Colors entry 167
 ~, backup files, and 57
 &, menu command text, in 127
 \a, menu command text, in 127

A

absolute align dialog box style 67
 absolute grid type 76
 Accelerator editor 139-151
 Command field 146
 Dialog Box pane, table of selections 142
 Key Value mode 142
 Menu editor, using with 143
 starting 144
 window panes 141
 accelerator keys
 View menu
 Paint editor 181
 accelerators 139-151
 case-sensitive 147
 changing 141
 undoing changes to 149
 compatibility issues, Windows, with 147
 copying 149
 creating 145
 Manual mode 148
 debugging 150
 defining 139
 deleting 149
 editing 141
 existing 144
 resource script 150
 flashing menu commands 149
 identifiers, checking for duplicates 150
 keys
 ASCII 146
 virtual 147
 menu commands, text in 127
 moving 149
 resource, creating 144
 resource script 150, 151
 sample project 151
 searching for duplicates 150
 tables 138
 viewing 141
 Add File to Project dialog box 42
 entering file search criteria 43
 adding a caption
 with Caption window 65
 with Window Style dialog box 65
 address, Borland 6
 Airbrush Shape command 193
 Airbrush tool 173
 brush shape, choosing 193
 size of painted area 173
 styles, Tools palette 178
 Align Controls dialog box 82, 83
 aligning
 controls 75
 with accelerators 83
 with short cut keys 83
 with the Alignment palette 84
 scroll bars 90
 selected area, Paint editor 196
 text, in Paint editor 191
 alignment options
 horizontal 83
 vertical 84
 Alignment palette 84
 Align menu 175
 All option (identifiers) 52
 ampersand, menu command text, in 127
 angles, drawing lines at 175
 ANSI character set 95, 99
 mapping fonts to 247
 mapping to 255

applications *See also* executable files
binding resources to 24-27

Array dialog box 85, 86

ASCII

- hex codes, strings

- parsed as 159, 261

- keys vs. virtual keys 146

Attributes options

- bitmaps 233, 236

Auto Check Box button 89

AUTOEXEC.BAT 6

Auto Radio Buttons 89

Auto 3-state buttons 89

B

background colors 168, 186

- eraser, and 172

backup files 57

BG indicator

- Colors palette 173

- Eraser tool 172

bibliography 6

binary files

- bitmaps, saving in 234

- cursors, saving in 224

- font 21

- icons, saving in 202

- identifiers, and 35

binding resources 21

BITMAP entry 234, 235

bitmapped images

- adding text to 175, 190-193

- color

- customizing 188-190

- number of, choosing 184-185

- copying 170

- deleting 170

- duplicating 170

- erasing 172

- filling with color 174

- grid overlay 182

- inverted areas 186

- changing color of 190

- loading 165

- painting 170-178

- resizing 181

- selecting areas of

- irregular 171

- rectangular 170

- style selections 178

- transparent areas 186

- changing color of 190

- viewing 178-180

- multiple views 178

- updating both 198

- zoomed 177

- zooming 180

bitmapped resources 167, *See also* specific

- resource type

- compiler differences in 265

- fonts vs. bitmaps 241, 242

- memory, and 242

- Paint editor support 167

- types 165

bitmaps 16, 233-239

- binary format

- saving in 234

- colors

- customizing 188

- number of

- changing 185

- choosing 235, 237

- compressing 238

- creating 234

- defining 242

- file type 21

- images

- creating 234-235

- customizing 236-238

- display options 233

- editing 235

- identifying in Project window 235

- inverted color areas 233

- loading 235

- resizing 233

- saving 238-239

- transparent attributes 233

- loading 234

- multiple, as fonts 242

- resizing 237

- resource scripts 236

- saving

- as resource scripts 234

- in binary files 238
 - sizing 235
 - storing 238
 - testing 239
- black-and-white images 218
- black frame static control 96
- Black Frame tool 72
- black rectangle control 72
- black rectangle static control 96
- .BMP files 21
- border, specifying a control 77
- borders *See* frames
- Borland, contacting 5
- Break Before options (menus) 119
- Brush Shape dialog box 193
- buttons *See also* specific type
 - controls, changing 88
 - customized 89
 - types 88
 - auto check box 89
 - auto radio 89
 - auto 3-state 89
 - check box 89
 - default push button 88
 - group box 89
 - owner draw 89
 - push button 88
 - radio button 89
 - 3-state 89
 - user 89
- Button Style dialog box 88
- By File command 38
- By Type command 39

C

- C programming language
 - defines 49
 - escape sequences 159
 - header files 22
- caption
 - adding a control 76, 77
 - adding to a dialog box 65
 - on title bar, displaying 66
- Caption field 97
- Caption window 77
- Case insensitive option (edit text controls) 94

- case-sensitive accelerators 147
- centering text in static controls 96
- CGA Resolution command 205, 226
- Change button 53
- Character options (fonts) 246-247
- character sets 249
 - ANSI 95, 99
 - defining 245, 247
 - mapping fonts to 247
 - mapping to 255
 - OEM 99
- character strings, in user-defined resources 261
- character underlining
 - in dialog boxes 96
 - in menus 127
- Character Width dialog box 247-248
- Check Box button 89
- check box dialog control 71
- Check Box tool 71
- Check Dup Keys command 150
- Check Duplicates command 131
- Checked option (menus) 119
- child windows *See also* controls
 - dialog boxes as 266
 - protecting client area of 68
- class, changing a control 78
- client area
 - protecting 68
 - restricting drawing to 68
- clip children dialog box style 68
- clip siblings dialog box style 68
- Clipboard
 - copying resources with 57
 - duplicating bitmapped images 170
 - Paste command, and 170
- Close box 67
- color images
 - compressed 238
 - noncompressed 238
- color options
 - Airbrush tool 174
 - Empty Frame tools 176
 - Filled Frame tools 177
 - Line tool 175
 - Paintbrush tool 173
 - Paint Can tool 174
 - Text tool 175

- colors 184-190
 - background 168, 186
 - eraser, and 172
 - bitmap 184-186, 235, 237
 - customizing 188
 - changing 189
 - cursor
 - inverted 226
 - transparent 226
 - editing 188-189
 - filling 174
 - foreground 168, 185
 - eraser, and 172
 - icon
 - inverted, changing 206, 227
 - transparent, changing 227
 - icons 185
 - customizing 188
 - format of 211
 - inverted 206
 - testing 208
 - transparent 206
 - changing 206
 - inverted 186
 - changing 190
 - leaking 174
 - lines 172, 175
 - number of, changing 185
 - predefined 198
 - text, Paint editor 191
 - transparent 186
 - changing 190
- Colors option (bitmaps) 237
- Colors palette 184, 188
 - BG indicator 173
 - displaying 187
 - editing colors 188-189
 - Default button 190
 - System button 190
 - FG indicator 173
 - Eraser tool 172
 - hiding 187
 - index 188
 - showing 187
- columns
 - dialog controls in 73
 - displaying text in list boxes in 92
 - new, in menus 119
- combo boxes 97
 - attributes
 - autohorizontal 99
 - integral height 99
 - OEM conversion 99
 - sorted 99
 - vertical scroll 99
 - customizing 97
 - dialog control 71
 - owner drawing options 98
 - resizing 99
 - scroll bars in 99
 - scrolling text in 99
 - sorting in 99
 - types
 - drop down 98
 - drop down list 98
 - simple 98
- combo boxes tool 71
- Combo Box Style dialog box 97
- Combo Box tool 71
- command-line options 10
- commands *See also* specific command
 - adding to accelerators 146
 - duplicating 139
 - facilitating 16
 - menu
 - accelerator keys and 139
 - menu resource
 - accelerator text 127
 - adding to 123
 - changing items 126
 - creating items 126
 - deleting 130
 - disabling 119
 - editing the text 127
 - enabling 119
 - graying 119
 - identifiers 127
 - inserting 124
 - pop-up 128
 - adding 129
 - deleting 130
 - right-justifying 127
 - tabs in 127
- Paint editor, viewing descriptions of 183

- resource script 41
- toggled 119
- viewing descriptions of 28
- comments, inserting in resource scripts 42, 151
- compiler differences 265
 - #defines, parameterized 266
 - hexstring data type 265
 - in bitmapped resources 265
 - in numeric constant expressions 265
 - RCDATA resources, references to files in 266
- compressed files 238
- Compression option (bitmaps) 238
- CompuServe Forum, Borland 5
- CONFIG.SYS 6
- configuration options
 - backups 57
 - executable save 56
 - include path 56
 - multi-save 56
 - .RES save 56
 - text editor 56
 - undo levels 55
- constants
 - declarations 23, 49, 77
 - defining 22
 - viewing 39
- context-sensitive Help 12
- control attributes
 - border 77
 - disabled 77
 - group 77
 - tab stop 77
- control class, changing 78
- control ID *See also* identifiers
 - identifier as 77
- Control menu 67, 72
- controls *See also* specific control type
 - adding 72
 - caption to 76
 - multiple copies 72
 - in rows or columns 72
 - scroll bars to 77
 - to a selected group 74
 - with Control menu 72
 - aligning 82
 - center of dialog box 84
 - horizontally in center of sizing frame 84
 - horizontally to the left side 84
 - horizontally to the right side 84
 - in rows and columns 85, 86
 - vertically along top 84
 - vertically at bottom 84
 - vertically in center of dialog box 84
 - vertically in center of sizing frame 84
 - with a grid 75
 - with accelerators 83
 - with short cut keys 83
 - assigning a control ID to 77
 - black rectangle 72
 - canceling placement of 72
 - check box 71
 - combo box 71, 97
 - customizing 97
 - coordinates of 74
 - custom 72, 78, 99
 - adding 100
 - displaying 100
 - installing 100
 - deleting one in a group of selected 74
 - dialog 70
 - defined 61
 - disabling 77
 - drawing border around 77
 - edit text 71, 93
 - customizing 93
 - editing 73
 - groups 81
 - group box 71
 - grouping 77, 80
 - groups, editing 81
 - horizontal scroll bars 71
 - iconic static 72
 - list boxes 71, 90
 - columns in 92
 - multiple selection in 92
 - owner drawing options 91
 - resizing 92
 - sorting in 92
 - modifying 73, 76
 - moving 74
 - placing 72
 - push button 71
 - radio button 71
 - reordering 80

- resizing 74
 - multiple 85
- scroll bars 90
 - customizing 90
- selecting 74
- selecting multiple 73, 81
- spacing equally 84
- specifying height of 75
- specifying width of 75
- static 95
- testing 101
- text static 71
- types 70-71
- vertical scroll bars 71
- white frame 72
- control statements, generating 106
- control tools 70
- conventions
 - menu commands 5
 - typographic 5
- coordinates
 - pixel 228
 - in bitmaps 183
 - specifying for controls 74
- Copy command 149
- copying resources between projects 57
- .CUR files 21
 - creating new cursors 224
- cursors 221-231
 - active area 225, 228-229
 - binary format, saving in 224
 - creating 222-224
 - customizing 225-227
 - designing 225-226
 - editing 225
 - file type 21
 - inverted colors 226-227
 - loading 165
 - resource script 225, 231
 - saving 222, 223, 229-231
 - as resource scripts 223
 - in binary files 230
 - storing 234-235
 - testing 222, 229
 - transparent colors 226-227
 - zooming 226
- custom classes, assigning to a dialog box 69

- custom control libraries 100
- custom controls 72, 78, 99, *See also* controls
 - adding 100
 - displaying 100
 - installing 100
 - types
 - application-specific 99
 - installable 99
- customer assistance 6
- customizing dialog boxes 64
- Cut command 149

D

- data
 - adding to user-defined resource 260
 - binary, bitmaps saved as 44
 - types, hexstring 265
- debugging
 - accelerators 150
 - menu resources 131
- decompiling resources 21
- Default button, Edit Colors dialog box 190
- Default push button 88
- #defines 22, 49
 - control IDs, and 77
 - identifiers stored as 50
 - parameterized 266
 - viewing 39
 - Windows 52
- defining dialog boxes 64
- Delete Stringtable Item command 160
- developing Windows programs 6
- dialog box coordinates 67
- dialog boxes 14, 61-113
 - adding a caption to 65
 - as child windows 266
 - as resources 14
 - assigning a custom class to 69
 - comparing two 104
 - controls *See* controls
 - creating 62
 - customizing 64
 - defined 61
 - defining 64
 - editing 64
 - file type 21
 - fonts in 68

- frame styles 66
 - making them movable 68
 - making them resizable 67
 - menus and 69
 - modal 67
 - moving 64
 - resizing 64
 - sample project 107-113
 - saving 102
 - selecting 64
 - storing 102
 - Style 73, 76
 - testing 101
 - text, entering in 71, 93
 - viewing two 104
 - Dialog Box pane
 - accelerator editor 142
 - Menu editor 118
 - dialog box styles 67
 - absolute align 67
 - clip children 68
 - clip siblings 68
 - horizontal scroll 67
 - local edit 67
 - maximize box 67
 - minimize box 67
 - modal frame 68
 - no idle messages 68
 - system menu 67
 - system modal 67
 - thick frame 67
 - vertical scroll 67
 - dialog control class, changing 78
 - dialog controls *See* controls
 - Dialog editor 61-113
 - Alignment palette in 63
 - Caption window in 63
 - customizing the 104
 - display options 100, 106
 - selection border options 105
 - selection options 106
 - starting 62
 - status line 63, 101
 - units of measurement 105
 - Tools palette in 63, 108
 - Dialog editor modes 70
 - dialog units 105
 - defined 75
 - dimming controls 77
 - directories, paths 36
 - Directories list box 36
 - Disabled options, menus 119
 - disabling controls 77
 - disabling menu commands 119
 - Discardable memory option 48
 - display drivers 185
 - display options in Dialog editor 100
 - .DLG files 21, 102
 - DLG Resource Script entry 103
 - DLL files 21
 - DLLs 99
 - changing resources in 58
 - DOS version numbers 6
 - dotted rectangles, zoomed images and 181
 - draft display option 106
 - drawing lines
 - angled 175
 - free-form 172
 - straight 175
 - Draw on Both Images option (Paint editor) 198
 - drivers, display 185
 - drop-down list
 - text always displayed in 98
 - user expandable 98
 - drop-down menus *See* menus
 - drop shading 207, 216
 - .DRV files 24
 - Duplicate command 170
 - Duplicate Control dialog box 73
 - Duplicate tool 73
 - Dynamic Link Library *See* DLL
- ## E
- Edit As Text command 40-41, 260
 - Edit Background Color command
 - cursors 227
 - icons 206
 - Edit Color commands 188
 - Edit command 40-41, 260
 - edit control 71
 - Edit Foreground Color command
 - cursors 227
 - icons 206
 - Edit Icon button 96

- Edit Image command 204, 211
- editing dialog boxes 64
- edit mode
 - return to 80, 101
- Editor Options command 198
- editors *See also* resource editors
 - selecting 40
 - text 19-20
 - internal, using 41-42
- Edit | Select All command 170
- edit text controls 93
 - aligning text in 93
 - allocate to local heap 67
 - case of text in 94
 - combo box control, in 97
 - converting text to OEM in 95
 - customizing 93
 - highlighting selected text in 95
 - number of lines in 94
 - password protection in 95
 - scrolling text in 94
- Edit Text Style dialog box 93
- Edit Text tool 71
- Edit Transparent and Inverted Colors dialog box 206
- Edit Transparent Color command 227
- Ellipse tool
 - empty frame 176
 - filled frame 177
- Enabled option (menus) 119
- enabling menu commands 119
- END POPUP statement 125
- Eraser tool 172, 186
- error messages, in string tables 155
- escape sequences, C type 159
- EVGA drivers 185
- executable files 21
 - changing resources in 58
 - saving resources in 56
 - command-line option 11
 - Preferences dialog box 57
- Exit command 11
- exiting
 - Help 12
 - Resource Workshop 11
- expressions, numeric constant, compiler differences in 265

- extensions, file
 - nonstandard 35
 - standard 24

F

- FB indicator, Colors palette 172
- FG indicator
 - Colors palette 173
 - Eraser tool 172
- fields, String editor 159, 164
- file combo boxes, entering file search criteria 43
- File Type option (Open File dialog box) 35
- file types 20-22
 - .BMP 21
 - .CUR 21
 - creating new cursors 224
 - .DLG 21, 102
 - DLL 21
 - .DRV 24
 - executable 21
 - .FNT 21
 - .FON 22
 - .ICO 21
 - creating new icons 202
 - .RC 20
 - .RES 20
- files *See also* specific file name; type
 - backing up 57
 - bitmap, references to in Project window 239
 - compressing 238
 - creating, by adding to project 44
 - cursor, references to in Project window 231
 - directories, paths 36
 - executable
 - changing resources in 58
 - saving resources in 56
 - extensions
 - nonstandard 35
 - standard 24
 - external, storing user-defined resources in 262
 - font, references to in Project window 252
 - formats 20
 - header 22
 - icon, references to in Project window 210
 - identifier 50
 - adding to projects 50

- C header 22
 - creating 44
 - Pascal include 23
 - Pascal units 23
- names, new user-defined resource 260
- RCDATA resources, referencing in 266
- renaming 54
- saving 53, 57
 - bitmap resources as 239
 - cursor resources as 231
 - font resources as 252
 - icon resources as 210
 - resources in 54
 - types, choosing 33
 - viewing resources by 38
- Filled Ellipse tool 194
- Filled Rectangle tool 194
- Filled Rounded Rectangle tool 194
- filling image with color 174
- First option (fonts) 247
- Fixed memory option 48
- flash feature 149
- .FNT files 21
 - creating 253
- .FON files 22
 - creating 253
- Font command 175, 190, 192
- FONT entry 244
- Font Header Information dialog box 249
- Font menu 251
- font size, Select Font dialog box 68
- Font Size command 245, 248, 251
- Font Size Information dialog box 245, 248
- fonts 18, 241-256, *See also* bitmapped images
 - assigning to text, in Paint editor 192
 - attributes, setting 250, 251
 - Attributes options 250
 - Average Width option 246
 - binary vs. runtime 21
 - bitmaps, multiple 242
 - Break option 247
 - changing 245
 - character options 247
 - character sets
 - defining 245
 - mapping 255
 - copyright information 249
 - Copyright option 250
 - creating 243
 - customizing 244-252
 - Default option 247
 - defining 242
 - deleting an image 251
 - describing 249
 - Device option 250
 - dialog boxes, in 68
 - editing 244-249
 - editor, status line in 255
 - Face Name option 250
 - file types
 - binary 21
 - runtime 22
 - header options 250
 - Height option 246
 - identifying in Project window 244
 - images 245
 - Last option 247
 - loading 165, 245
 - mapping character sets to 247
 - Maximum Width option 246
 - outline 241
 - picture 241-243, 247
 - raster 241
 - resizing 246
 - resource script 243, 245
 - sample project 254-256
 - saving 252-253
 - as resource scripts 243
 - in binary files 252
 - sizing 245, 251
 - options 246
 - storing 243
 - Stretch Current Chars option 246
 - testing 254
 - variable-width 246-248
 - viewing 255
- Font Version option (font headers) 250
- foreground colors 168, 185
 - eraser, and 172
- Format option (bitmaps) 238
- format specifiers
 - hexadecimal 159, 261, 265
 - octal 261

- formats, storage
 - cursors 234-235
 - icons 201-203
- frame style
 - border 66
 - caption 66, 67
 - dialog frame 66
 - no border 66
- Frame tools
 - color options 177
 - empty, color options 176
 - filled 177
- frames 195
 - painting 176
- free-form lines 172
- free-form patterns 173-174
- functions
 - Paint editor 167
 - program, and cursors 221

G

- Generic Control Style dialog box 78
- grabbing images 177
- graphics *See* bitmapped images
- Graphics Device Interface (GDI) 257
- gray frame static control 96
- Grayed option (menus) 119
- graying controls 77
- graying menu commands 119
- gray rectangle static control 96
- grid
 - aligning controls with 75
 - displaying 75
 - Paint editor screens, on 182
 - type
 - absolute 76
 - relative 76
- Grid on zoomed images option 198
- Group Box buttons 89
- group box dialog control 71
- Group Box tool 71
- grouping controls 77, 80
 - with Set Groups menu command 80
 - with Set Groups tool 80
- groups, moving within control 80

H

- Hand tool 177
- hardware requirements 2
- Header command 249, 251
- header files (C) 22
- headers (fonts) 249
- Help
 - accessing 12
 - Borland technical support 6
 - context-sensitive 12
 - cursor 12
 - exiting 12
 - resource scripts 41
- Help Break option 119
- hexadecimal format specifiers 159, 261, 265
- hexstring data type 261, 265
- Hide Palette command 187
- Hide Toolbox command 169
- Home Budget icon 213-219
- horizontal alignment options 83
- horizontal lines *See* lines
- horizontal pixel units 228
- horizontal scroll bar dialog control 71
- horizontal scroll bars *See* scroll bars
- Horizontal Scroll Bar tool 71
- horizontal scroll dialog box style 67
- Horizontal Size option (controls) 86
- hot spots 225
 - setting 228-229
 - testing 229

I

- .ICO files 21
 - creating new icons 202
- ICON entry 201, 203
- iconic static control 72
- Iconic Static Text tool 72
- icons 17, 199-219, *See also* bitmapped images
 - attributes, setting 184-185
 - binary format, saving in 202
 - color options 185, 211
 - customizing 188-189
 - colors
 - customizing 188
 - number of, changing 185
 - creating 200-203, 214

- black-and-white 218
- three-dimensional 207, 216
- customizing 204-207
- deleting
 - image 212
 - resource 212
- designing 204
- dialog boxes, in
 - editing a static control 96
 - resource ID 97
- display options 205, 211
- drawing 215, 217
 - sample calculator 214
- editing 200, 204
- erasing 216
- file type 21
- ICON window 201
- images, adding to 211
- inverted colors 205-207
- loading 166, 201
- multiple images 204
 - adding 211
 - creating 218
- resource script 213
- sample project 213-219
- saving 201, 209
 - as resource scripts 203
 - in binary files 210
- starting Paint editor 166
- storing 201-203
- testing 208-209
- transparent colors 205-207
- zooming 205
- Icon window 166, 204
- ID Source and Value fields 158-160
- identifiers 22, 49-53
 - binary files, and 35
 - checking for duplicates
 - in accelerators 150
 - in menus 131
 - creating 50
 - new resource type 258
 - editing 51
 - files 50
 - adding to projects 50
 - C header 22
 - creating 44
 - Pascal include 23
 - Pascal units 23
 - include path option 56
 - incremented 164
 - length of 53
 - MAKEINTRESOURCE macro 46
 - menu commands 119
 - adding to accelerators 146
 - menus 127
 - storing 28, 50
 - string 158
 - string table 162, 163, 164
 - changing 161
 - typesetting 46
 - user-defined resources 258, 260
 - viewing 39
 - virtual keys 147
 - Windows, displaying 52
- Identifiers dialog box 51
- IDs *See also* identifiers
 - menu commands
 - adding to 119
 - adding to accelerators 146
 - resource 46
 - string table 158
 - numbering 157
- images *See* bitmapped images, zoomed images
- include files (Pascal) 23
- Include Path option
 - command-line switch 11
 - Preferences dialog box 56
- Initial State options (menus) 119
- Install a New Control Library dialog box 100
- installing Resource Workshop 9
- integer IDs *See* identifiers
- internal text editor *See* text editor
- inverted areas 186
 - colors, changing 190
 - cursors 226-227
 - icons 205-207, 226
- Inverted option 186
- Invert Menu Item option (accelerators) 149
- invert menu item *See* flash feature
- Item ID (menus) 119, 127, 131
- Item Text (menus) 119
- Item Type (menus) 119

J

justifying control text 90

K

keyboard accelerators 15, *See also* accelerators;
accelerator tables

keys

ASCII 146

defining as accelerators 139, 147

text editor 261

virtual 147

Key Value command 148

Key Value mode, tab key and 142

L

line patterns *See* patterns

Line tool 175, 195

styles, Tools palette 178

Line tool (Paint editor) 228

lines

color options 172, 175

new

in menu bar 119

in menus 119

painting

angled 175

free-form 172

straight 175

styles, choosing 195

List Box options 92

List Box Style dialog box 91

List Box tool 71

list boxes

adding to dialog boxes 90-92

columns in 92

combo box control 97

dialog control 71

multiple selection in 92

resizing 92

scrolling horizontally in 92

selecting multiple items in 92

sorting in 92

tab stops in 92

Load on Call memory option 48

local edit dialog box style 67

Lower Case option (edit text controls) 94

M

mailing address, Borland 6

MAKEINTRESOURCE macro 46

Manual mode, creating accelerators in 148

maximize box dialog box style 67

Maximize buttons

adding 67

bitmaps, as 233

memory

bitmapped resources, and 242

options 47

Discardable 48

Fixed 48

Load on Call 48

menus 134

Moveable 48

Nondiscardable 48

Pure 49

Memory Options command 47

Menu Bar Break option (menus) 119

menu bars

new line in 119

Paint editor 167

Menu Break option (menus) 119

Menu editor 115-138

Accelerator editor, using with 143

activating 121, 122

Break Before option 119

Dialog Box pane 118

Item ID option 119

Item Text option 119

Item Type option 119

Menu Break option 119

No Break option 119

Outline pane 117

reconfiguring window 120

Undo command 123

View as Pop-up command 120

window panes 117-120

Menu Editor screen 117

Menu menu 123

menu resource 69

menus 15, 115-138

accelerator keys and 139

adding new statements 122

changing items 126

command conventions 5

- commands 123
 - accelerator text 127
 - adding to 123
 - adding to accelerators 146
 - disabling 119
 - editing the text 127
 - enabling 119
 - graying 119
 - inserting 124
 - pop-up 128
 - adding 129
 - deleting 130
 - right-justifying 127
 - tabs in 127
- copying 123
- creating 121
- creating new items 126
- customizing 122
- debugging 131
- dialog boxes and 69
- editing 122
 - text version of 133
- END POPUP statement 125
- flashing 149
- Help break in menu bar 119
- identifiers 127
 - checking for duplicates 131
- items 123
- memory options 134
- menu separators 130
- moving 123
- new column in 119
- new line in 119
 - menu bar 119
- pop-up commands 123
- POPUP statement 125
- resource script 133
- sample project 135
- separators 130
- statements, deleting 130
- test, displaying as pop-up 120
- testing 131
- messages, Paint editor tool status 183
- metafiles 257
- Microsoft Resource Compiler compatibility 78
- Microsoft Windows *See* Windows
- minimize box dialog box style 67

- Minimize buttons
 - adding 67
 - bitmaps, as 233
- minimized windows 17, 199
- modal frame dialog box style 68
- modeless dialog box 52
- modes
 - Dialog editor, in 70
 - edit 79, 80
 - selection 72, 73, 81
 - test 101
- monochrome images, icon 218
- Moveable memory option 48
- moving dialog boxes 64
- Multi-Save options
 - command-line switch
 - executable file 11
 - .RES file 11
 - Preferences dialog box 56
- multiple controls
 - placing copies of 72
 - selecting 73, 81
- multiple dialog controls
 - placing in rows or columns 72
 - selecting 73
- MYPROJ.H 163
- MYPROJ.RC 136

N

- New Bitmap Attributes dialog box 235
- New button 53
- New command 32, 42
- New Custom Control dialog box 100
- New Icon Image dialog box 202, 203
- New Identifier dialog box 47, 50, 51
- New Image command 166
 - icons 211
- New Item command 163
- New Menu Item command 126
- New Pop-up command 129
- New Project dialog box 32
- New Resource dialog box 44, 62
- New Resource Type dialog box 258
- New Separator command 130
- No Break option (menus) 119
- no idle messages dialog box style 68
- NOINVERT command 150

- Nondiscardable memory option 48
- nonstandard resources *See* user-defined resources
- nonvariable-width fonts 247
- normal display option 106
- Notepad editor (Windows) 41
- null pen width 195
- numbering string tables 157
- numeric constant expressions, compiler differences in 265
- numeric IDs *See* identifiers
- numeric values, notation for, in user-defined resources 261

O

- octal format specifiers 261
- OEM character set 95, 99
- online Help, accessing 12
- Open command 42
- Open File dialog box 34
- outline fonts 241
- Outline pane
 - Accelerator editor 141
 - Menu editor 117
- outlines *See* frames
- Owner Draw buttons 89
- Owner Drawing options 91
 - combo boxes, in 98

P

- Paintbrush tool 173
 - brush shape, choosing 193
 - styles, Tools palette 178
- Paint Can tool 174
- Paint editor 165-198
 - Airbrush tool 173
 - aligning selected area 196
 - Colors palette 184, 188-190
 - Ellipse tool
 - empty frame 176
 - filled frame 177
 - Eraser tool 172
 - functions 167
 - Hand tool 177
 - Line tool 175

- multiple views 178
- options 198
- Paintbrush tool 173
- Paint Can tool 174
- Pen tool 172
- Pick Rectangle tool 170
- Rectangle tool
 - empty frame 176
 - filled frame 177
- resizing a selected area 196
- Rounded Rectangle tool
 - empty frame 176
 - filled frame 177
- Scissors tool 171
- scroll bars 182
- starting 165
 - from ICON window 166
- status line 183
- stretching a selected area 196
- style selections 178
- Text tool 175
- tools 170-178
- Tools palette 168-178
 - functions 167
- windows 178-182, 245
 - icons 201
- Zoom tool 171

palette index, Paint editor 183

palettes

- Alignment 84
- Colors 184
 - index 188
- Tools
 - Dialog editor 70
 - Paint editor 169

panes *See* window panes

Pascal

- constant declarations 49
- include files 23
- units 23

Paste command 123, 149

- Clipboard, and 170

Paste Resource dialog box 58

Pattern command 194

patterns 194

- Filled Frame tools 177
- Paint editor 194

- painting
 - with airbrush 173
 - with paintbrush 173
 - styles, Tools palette 178
- Pen tool 172
 - line style 195
- Pen Width command 195
- Pick Rectangle tool 72
 - Paint editor 170
 - selection mode and 73
- picture fonts 241-243, 248
- pixel coordinates, counting 183, 228
- pop-up commands, in menus 128
- pop-up menus *See* menus
- POPUP statement 125
 - deleting 130
- predefined colors 198
- preferences
 - backups 57
 - executable save 56
 - include path 56
 - multi-save 56
 - .RES save 56
 - text editor 56
 - undo levels 55
- Preferences dialog box 55
- program functions *See* functions
- programming in Windows 6
- programs *See* executable files
- Project window 27
 - files
 - bitmap, references to 239
 - cursor, references to 231
 - font, references to 252
 - icon, references to 210
 - resources
 - displaying by file 38
 - displaying by type 39
 - seeing details 40
 - selecting 40
 - types, displaying unused 40
- projects 31-55
 - adding existing resources to 42
 - adding resource types to 258
 - adding resources to 42-45
 - compiling 53
 - copying resources between 57

- creating new 32
- opening 34
- overview, sample 24
- renaming 54
- sample *See* sample projects
- saving 53, 102
 - working with 27-28
- prompts, in string tables 155
- Pure memory option 49
- push button dialog control 71
- Push Buttons 88
- Push Button tool 71

Q

- quitting
 - Help 12
 - Resource Workshop 11

R

- radio button dialog control 71
- radio buttons 89
- Radio Button tool 71
- raster fonts 241
- .RC files 20
- RCDATA resources, references to files in 266
- RCDATA resource type 262
- Rectangle tool
 - empty frame 176
 - filled frame 177
- Redo command 149
 - Menu editor 123
- Redo feature 28
- Redo option 55
- relative grid type 76
- Rename command 45
- renaming
 - projects 54
 - resources 45
- reordering controls
 - with Set Order menu command 81
 - with Set Order tool 81
- .RES files 20
 - saving resources in
 - command-line option 11
 - Preferences dialog box 56
- resizable dialog boxes 67

- Resize Current Bitmap option 237
- resizing a selected area, in Paint editor 196
- resizing dialog boxes 64
- resource compiler 265
 - files 20, 24, 27
 - Windows 265
- resource editors 41
 - Accelerator editor 139-151
 - Dialog editor 61-113
 - Menu editor 115-138
 - Paint editor 165-198
 - selecting 40
 - String editor 155-164
- Resource Memory Options dialog box 48
- Resource Rename dialog box 46
- resource scripts 41
 - accelerators 150, 151
 - bitmaps 234
 - editing 236
 - saving as 234
 - comments, inserting 151
 - comments in 42
 - creating 261
 - cursors 230
 - editing 225, 231
 - saving as 223
 - dialogs 105
 - example 110
 - fonts 253
 - editing 245
 - format specifiers 159
 - help with 41
 - icon
 - editing 213
 - saving as 203
 - linking 266
 - menus 133
 - saving 44
 - storing bitmapped resources as 265
 - string table 160, 162
- Resource Workshop, features 1
- resources 13
 - accelerators 139-151
 - accessing by ID 46
 - adding new files to 44
 - adding to a project 42-45
 - as files 42
 - new 44
 - binding 21
 - changing in executable files 58
 - compiling 20, 27, 53, 265
 - copying between projects 57
 - creating
 - new resources 44
 - new types 258
 - decompiling 21
 - deleting 49
 - Discardable 48
 - displaying
 - by file 38
 - by type 39
 - in Project window 39
 - editing 19
 - file types
 - .BMP 21
 - .CUR 21
 - .DLG 21
 - DLL 21
 - executable 21
 - .FNT 21
 - .FON 22
 - .ICO 21
 - .RC 20
 - .RES 20
 - Fixed 48
 - fonts vs. bitmaps 241
 - identifying 22
 - IDs, assigning to 46
 - Load on Call 48
 - memory options 47
 - menus 115-138
 - Moveable 48
 - naming 22
 - Nondiscardable 48
 - organizing 28
 - Preload 48
 - Project window, seeing details 40
 - Pure 49
 - RCDATA 262
 - references to files in 266
 - renaming 45
 - reusing 54
 - saving 53
 - command-line options 11

- Preferences dialog 56
- selecting 40
- separate from program code 14
- types of 14-18
 - unused, displaying 40
- user-defined 18, 257-263
 - adding to project 258
 - deleting 263
 - editing 260
 - metafiles and 257
 - testing 262
- reusing resources 54
- RGB values, Paint editor 183, 189
- right-justifying text, in menu commands 127
- right mouse button, adding controls with 72
- Rounded Rectangle tool
 - empty frame 176
 - filled frame 177
- rows, dialog controls in 73
- runtime font files 22
- RWPDEMO.RC 107, 166, 254
- .RWS files 53

S

- sample project overview 24
- sample projects
 - accelerator 151
 - dialog boxes 107-113
 - fonts 254-256
 - icon 213-219
 - menu 135
 - minimizing 199
- Save File As command 54
- Save Project command 53
- Save Resource As command 54
- Save with Default Device Colors option 189, 198
- saving resources 53
 - in executable files
 - command-line options 11
 - Preferences dialog 56
 - in .RES files
 - command-line options 11
 - Preferences dialog 56
- Scissors tool 171
- screen units 105
- scripts *See* resource scripts
 - fonts 243
- Scroll Bar option 77
- scroll bars 90
 - adding to frame 67
 - aligning 90
 - as bitmaps 233
 - customizing 90
 - in controls 77
 - in list boxes 91
 - Paint editor 182
- Scroll Bar Style dialog box 90
- search criteria, entering in combo boxes 43
- segments, string 157
- Select Font dialog box 68, 191, 192
- selected areas, Paint editor
 - aligning 196
 - resizing 196
- selecting an entire image 170
- selection border 105
- selection mode 72, 73
- selection options 106
- separators, menu 130
- Set Bitmap Attributes dialog box 236
- Set Grid Attributes dialog box 75
- Set Groups tool 80
- Set Hot Spot command 228
- Set Order tool 81
- Set Paint Editor Options dialog box 198
- Set Pattern dialog box 194
- Set Pen Style dialog box 195
- shading *See* drop shading
- Show Identifiers command 39
- Show Items command 40
- Show Palette command 187
- Show Predefined option (Identifiers dialog box) 52
- Show Resources command 39
- Show Toolbox command 169
- Show Unused Types command 40
- siblings, protecting child window of 68
- Single File option (Identifiers dialog box) 52
- Size and Attributes command 185
- Size Controls dialog box 85
- size of font, Select Font dialog box 68
- size options, controls for
 - horizontal 86

- vertical 86
- sizing frame 64, 74
 - stretching 84
- software requirements 2
- sorting in combo boxes 99
- source code *See* resource scripts
- Split Horizontal command 180
- splitting, Paint editor window 178
- Split Vertical command 180
- starting Resource Workshop 9
- static controls 71, 95
 - aligning text in 96
 - background color 96
 - black frame 96
 - black rectangle 72, 96
 - character underlining in 96
 - control ID of 77
 - displaying centered text 96
 - gray frame 96
 - gray rectangle 96
 - iconic 72, 96
 - left justifying text in 96
 - right justifying text in 96
 - text 71
 - white frame 72, 96
 - white rectangle 96
 - wordwrapping text in 96
- Static Style dialog box 95
- status line 28
 - Dialog editor 63, 101
 - Paint editor 183
 - Font resources 255
 - units of measurement on 105
- storage formats
 - cursors 234-235
 - icons 201-203
- straight lines *See* lines
- stretching selected area, in Paint editor 196
- String editor 155-164
 - activating 156, 163
 - fields 159, 160
 - starting 156
- string resources *See* string tables
- string segments 164
- string tables 16, 155-164
 - changing 160
 - closing 164

- deleting 160
- editing 160
 - text version of 160
- figure showing screen 157
- identifiers 162-164
 - changing 161
 - moving around in 160
 - naming 164
 - numbering strings 157
 - resource script 160, 162
- strings
 - character, in user-defined resources 261
 - text *See* string tables
- Style dialog boxes 76
- styles, Paint editor
 - airbrush 193
 - paintbrush 193
 - patterns 194
 - pen 195
 - selecting from Tools palette 178
- symbolic constants *See* identifiers
- System button, Edit Colors dialog box 190
- System menu 67
- system modal dialog box style 67
- system requirements 2

T

- Tab Set tool 79
- tab stops 77, 78
 - changing 79
 - control attribute 77
 - in list boxes 92
 - setting
 - with Set Tabs menu command 79
 - with Tab Set tool 78
 - with Tab Stop attribute 79
 - specifying controls as 78
- tabbing, and Key Value mode 142
- tables, accelerator *See* accelerator tables
- tabs, in menu command text 127
- technical support, Borland 6
- Test command (cursors) 229
- Test Dialog command 101
- Test Menu pane 120, 131
- Test Menu window
 - changing position 120
 - pop-up menu, displaying as 120

- test mode, exiting 101
- testing
 - bitmaps 239
 - cursors 229
 - dialog boxes 101
 - fonts 254
 - icons 208
 - menu resources 131
 - user-defined resources 262
- text
 - adding to bitmapped images 175, 190-193
 - aligning 90
 - Paint editor, in 191
 - case sensitivity 94
 - color options 175
 - editing 155
 - entering into dialog box 71
 - font and size, changing in Paint editor 192
 - menu commands, adding to 119, 137
 - Paint editor, color options 191
 - pop-up commands, adding to 129
 - resizing 175
 - strings *See* string tables
 - translating 155
- text editor
 - internal
 - selecting 40
 - using 41-42
 - keys 261
 - user-defined resources, using with 259
- Text Editor option (Preferences dialog box) 56
- Text menu 190
- text static control 71
- Text Static Control tool 71
- text strings
 - editing 160
 - listing 90
 - null-terminated 261
- text style, Select Font dialog box 68
- Text tool, Paint editor 175
- thick frame dialog box style 67
- three-dimensional icons 207, 216
- 3-State buttons 89
- tildes in file names 57
- tips, for new users 28
- title bar, displaying caption on 65
- toggles, menu commands 119
- tool
 - black frame 72
 - check box 71
 - combo box 71
 - custom control 72
 - edit text control 71
 - group box 71
 - horizontal scroll bar 71
 - iconic control 72
 - list box 71
 - push button 71
 - radio button 71
 - text static control 71
 - vertical scroll bar 71
 - white frame control 72
- tool palettes *See* palettes
- tools
 - Dialog editor 70
 - Paint editor 170-178
 - selecting 169
 - status messages 183
 - viewing current 183
- Tools palette 70
 - Dialog editor 70, 108
 - Paint editor 168-178
 - closing 169
 - functions 167
 - opening 169
 - selecting tools 168
- translating a user interface 58
- translating text 155
- transparent areas 186
 - colors, changing 190
 - cursors 226-227
 - icons 205-207, 226
- Transparent option 186
- type, viewing resources by 39
- typecasting resource IDs 46
- typefaces *See* fonts
 - Select Font dialog box 68
- Type options (dialog boxes) 97
- typestyles 249, 250, *See also* fonts
- typing accelerators 148
- typographic conventions 5

U

- underlining characters
 - in menus 127
 - in static controls 96

Undo

- command
 - Accelerator editor 149
 - Menu editor 123
- feature 28
- option 55
- tool 88

Undoing edits 88

Undo Size Group command 86

unfilled ellipse tool, line style 195

unfilled rectangle tool, line style 195

unfilled rounded rectangle tool, line style 195

unit files (Pascal) 23

Upper Case option (edit text controls) 94

User Buttons 89

user-defined resources 18, 257-263

- adding data to 260
- adding to project 258
- data formats 261
- deleting 263
- editing 260
- identifiers 257, 258, 260
- metafiles and 257
- storing in external files 262
- text strings, null-terminated 261

user interface, changing in an executable file 58

V

variable-width fonts, creating 247-248

vertical alignment options 83

vertical lines *See* lines

vertical pixel units 228

vertical scroll bar dialog control 71

vertical scroll bars *See* scroll bars

Vertical Scroll Bar tool 71

vertical scroll dialog box style 67

VGA drivers 185

video drivers 185

View as Pop-up command, Menu editor 120

View menu

- Menu editor 120
- Project window 38

zooming with 180

virtual keys 147

W

white frame static control 72, 96

White Frame tool 72

white rectangle static control 96

Widgets menu

- accelerators, creating 151-153
- creating 135-138

Width option (fonts) 246

window panes

- Accelerator editor 141-142

- Menu editor 117-120

Window Style dialog box 64, 66, 68

window types 66

- child 66

- iconic popup 66

- overlapped 66

- popup 66

Windows

- accelerators, compatibility issues 147

Clipboard

- duplicating bitmapped images 170

- Paste command 170

- using to copy resources 57

identifiers

- displaying 52

Notepad editor 41

programs, writing 6

resource compiler 265

software development 6

string table numbering 157

system requirements 2

VGA driver 185

windows

- Accelerator editor 141-142

- child, and dialog boxes 266

- ICON 166, 201

- Icon 204

- minimized 17, 199

- Paint editor 178-182, 245

- dotted rectangles in 181

- Project *See* Project window

wordwrapping text in static controls 96

WYSIWYG display option 106

X

X value (pixel coordinates) *183, 228*

Y

Y value (pixel coordinates) *183, 228*

Z

zoom commands *181*

 Zoom In *171*

 Zoom Out *171*

zoomed images

 grid overlay on *182*

 moving *177, 182*

 scaling *180*

zooming

 accelerators, with *181*

 airbrush, and *173*

 cursors, and CGA command *226*

 dotted rectangle, and *181*

 icons, CGA command, and *205*

 images

 entire *171, 180*

 part of *171, 180*

 paintbrush, and *173*

Zoom tool *171, 180*

**USER'S
GUIDE**

RESOURCE WORKSHOP

B O R L A N D

Corporate Headquarters: 1800 Green Hills Road, P.O. Box 660001, Scotts Valley, CA 95067-0001, (408) 438-5300. Offices in: Australia, Belgium, Canada, Denmark, France, Germany, Hong Kong, Italy, Japan, Korea, Malaysia, Netherlands, New Zealand, Singapore, Spain, Sweden, Taiwan and United Kingdom ■ Part # 14MN-RWS01-10 ■ BI 2434